Universidad de Buenos Aires

Facultades de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Maestría en Seguridad Informática

Tesis de Maestría

Título de la Tesis: Despliegue Preventivo de servicios web en contenedores docker basado en SecDevOps

Autor: Lic. Hurson Azuaga

Tutor: Ing. Juan Alejandro Devincenzi

Cohorte 2019

#### Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Tesis vigente y que se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

FIRMADO

Nombres y Apellidos: Hurson Daniel Azuaga Orrego. Número de Documento: 95836734.

#### Resumen

La CI/CD es un método para distribuir aplicaciones mediante la automatización en las etapas del desarrollo de aplicaciones; esto se consigue por medio de la integración de varias soluciones orquestado por un servidor de automatización.

El objetivo del siguiente trabajo es presentar un esquema de actualización continua para servicios web Spring Boot en contenedores docker como un componente más dentro de una CI; de forma a mitigar los problemas de seguridad dado por los componentes obsoletos que puedan existir.

El aporte de este trabajo consiste en integrar en un pipeline la tarea de pruebas de actualización dentro de una CI; de manera que al implementar un nuevo servicio web Spring Boot en un contenedor docker se implemente con una plataforma actualizada y probada. Para ello se expondrán una serie de herramientas ejecutadas por medios de pipelines en Jenkins.

#### Palabras Claves

Seguridad, Integración, Actualización, Vulnerabilidad.

Tał	ola	de	cor	nter	nido
IGN	ла	uc	CUI	ILCI	IIUU

Lista de Figuras	1
Agradecimientos	5
Nómina de Abreviaturas	6
Introducción	7
<ol> <li>Diseño y arquitectura de la seguridad en un ambiente CI/CD</li> <li>1.1 Entorno de Pruebas</li> <li>1.2 Entorno de Producción</li> <li>1.3 Entorno de Prueba con Jenkins</li> </ol>	<b>9</b> 9 10 11
2. Diseño y arquitectura del pipeline en la Cl_PruebasActualizacion.	tarea 12
<ul> <li>3. Pruebas de actualización en la organización.</li> <li>3.1 Pruebas de actualización y Desarrollo</li> <li>3.2 Pruebas de actualización y QA</li> <li>3.3 Pruebas de actualización y Seguridad</li> <li>3.3.1 Docker</li> <li>3.3.1.1 Docker File</li> <li>3.3.1.2 Docker Compose</li> <li>3.3.2 Trivy</li> <li>3.4 Pruebas de Actualización y Operaciones</li> <li>3.4.1 Jenkins</li> <li>3.4.1.1 Creacion de Tareas Jenkins</li> <li>3.4.1.2 Configuración del Plugin Build Pipeline</li> <li>3.4.1.3 Configuración de la tarea CI_PruebasActualizacion</li> <li>3.4.1.3.1 Lightning</li> <li>3.4.1.3.2 Lightning y Apache Jmeter</li> <li>3.4.1.3.1 Trivy y Jenkins</li> </ul>	<ul> <li>13</li> <li>14</li> <li>21</li> <li>28</li> <li>28</li> <li>30</li> <li>34</li> <li>38</li> <li>41</li> <li>42</li> <li>46</li> <li>48</li> <li>52</li> <li>57</li> <li>58</li> <li>62</li> </ul>
<ul> <li>4. Implementación de las pruebas de actualización</li> <li>4.1 Formas de implementar la tarea de actualización</li> <li>4.2 Ventajas de la actualización continua</li> </ul>	<b>65</b> 74 75
5. Conclusión	76
6. Bibliografía	78

## Lista de Figuras

Figura 1. Entornos de una CI/CD.

Figura 2. El Entorno de Pruebas y los datos.

Figura 3. El Entorno de Producción y los datos.

Figura 4. Entorno de Pruebas.

Figura 5. Pipeline DevOps [8].

Figura 6. Diseño del pipeline de Actualización.

Figura 7. Esquema de Pruebas Seguridad, QA, Operaciones y Desarrollo.

Figura 8 Página de descarga JAVA JDK 11.

Figura 9 Wizard de instalación JDK 11.0.11.

Figura 10 Página de descarga Apache Netbeans 12.3.

Figura 11 Archivo de instalación Apache NetBeans 12.3.

Figura 12 Comando sh para instalación de Apache NetBeans 12.3.

Figura 13 Página de creación para aplicaciones Spring Boot.

Figura 14 Building REST Services with Spring.

Figura 15 Proyecto Spring Boot Employee.

Figura 16 Clase EmployeeController.

Figura 17 Resultado de la compilación del proyecto Spring Boot.

Figura 18 Salida del proyecto Spring Boot al iniciar.

Figura 19 Respuesta del servicio.

Figura 20 Generación de Entorno de Prueba QA.

Figura 21. Página de descarga de Apache Jmeter.

Figura 22 Apache Jmeter.

Figura 23 Apache Jmeter opción configurar hilos.

Figura 24 Opción de Hilos Jmeter.

Figura 25 Configuración de una petición HTTP en Apache Jmeter.

Figura 26 Peticiones Http Get, Post, Put, Delete en Apache Jmeter.

Figura 27 Petición Get.

Figura 28 Petición Post.

Figura 29 Petición Put.

Figura 30 Petición Delete.

Figura 31 Resultado de las pruebas en Jmeter.

Figura 32 Proyecto Empleado.jmx.

Figura 33 Página de Instalación de Docker.

Figura 34 Versión de Docker instalada.

Figura 35 Arquitectura del Dockerfile para el pipeline de actualización.

Figura 36 Dockerfile con SO alpine en su versión latest.

Figura 37 Dockerfile con SO alpine en su versión producción alpine 3.10.

Figura 38 Construcción del Dockerfile con SO alpine en su versión producción alpine 3.10.

Figura 39 Esquema para archivo Docker Compose.

Figura 40 Organización de ficheros para el docker-compose.

Figura 41 Archivo docker-compose.

Figura 42 Salida de consola una vez ejecutado el docker-compose.yaml.

Figura 43 Contenedores en ejecución.

Figura 44. Esquema de funcionamiento de la herramienta Trivy.

Figura 45 Comandos de instalación de Trivy.

Figura 46 Versión del Trivy instalada.

Figura 47 Comando para analizar desde una imagen con Trivy.

Figura 48 Salida del comando trivy servicios\_app1.

Figura 49 Opción severity Trivy.

Figura 50 Opción severity CRITICAL Trivy.

Figura 51 Página de Instalación de Jenkins.

Figura 52 Comandos de instalación de Jenkins.

Figura 53 Comando para iniciar Jenkins.

Figura 54 Wizard de configuración de Jenkins desbloqueo.

Figura 55 Wizard de configuración de Jenkins selección de plugin.

Figura 56 Wizard de configuración de Jenkins instalación de plugin.

Figura 57 Wizard de configuración de Jenkins creación de la cuenta.

Figura 58 Página de inicio Jenkins.

Figura 59 Opción Nueva Tarea.

Figura 60 Creación de un trabajo en Jenkins.

Figura 61 Trabajos para la ejecución de los pipelines.

Figura 62 Menú Administrar Jenkins.

Figura 63 Opciones de System Configuration.

Figura 64 Build Pipeline Plugin.

Figura 65 Opción para agregar pipelines.

Figura 66 Creación de Pipeline.

Figura 67 Opciones pipeline TesisMaestria.

Figura 68 Configuración del primer trabajo a ejecutar en la CI.

Figura 69 Opción de Disparadores de Ejecución dentro de la tarea CI\_PruebasActualizacion.

Figura 70 Flujo de las tareas en la CI.

Figura 71 Configuración de Disparadores de Ejecuciones.

Figura 72 Estructura de directorios en repositorio GitHub.

Figura 73 Configuración de un proyecto GitHub.

Figura 74 Configuración del origen de código en Jenkins.

Figura 75 URL de conexión HTTPS Github..

Figura 76 Opción Ejecutar línea de comandos (shell).

Figura 77 Cuadro para insertar líneas de comandos.

Figura 78 Salida de la consola de Jenkins cuando construye los contenedores.

Figura 79 Contenedores en ejecución.Contenedores en ejecución.

Figura 80 Comando para clonar Lightning desde el repositorio Git hub.

Figura 81 Configuración xml para análisis con Lightning.

Figura 82 Configuración del archivo jmeter.properties.

Figura 83 Archivo PruebaJmLig.csv.

Figura 84 Comandos Lightning en Jenkins.

Figura 85 Log de Jenkins al ejecutar Lightning con errores encontrados.

Figura 86 Log de Jenkins al ejecutar Lightning con resultado exitoso.

Figura 87 Log de Jenkins al ejecutar Trivy.

Figura 88 Reporte de vulnerabilidades Trivy.

Figura 89 Comandos de ejecución para la tarea CI\_PruebasActualizacion.

Figura 90 Opción de Jenkins para construir la tarea.

Figura 91 Log de Jenkins al ejecutar la tarea CI\_PruebasActualizacion.

#### Agradecimientos

El desarrollo de esta tesis requirió bastante esfuerzo y muchísima dedicación, es por eso que agradezco inmensamente a mi familia mis hermanos, Marco Antonio Azuaga Orrego, Francisco Javier Azuaga Orrego y Sandra Raquel Azuaga Orrego, muy en especial a mi padre Hurson Azuaga Rios y mi madre Elena Juliana Orrego de Azuaga (+) que siempre me empujaron por el camino de la educación y me inculcaron valores que me sirven hasta hoy en dia en la vida.

A mi esposa Karina Del Rocio por estar a mi lado apoyándome a seguir adelante a pesar de las dificultades presentadas y brindarme apoyo emocional durante todos mis estudios de maestría. A mis amigos en especial a Carlos Aranda por ayudarme en los problemas personales cuando cursaba la maestría, al Prof Ing Hugo Pagola por orientarme en el ámbito profesional y darme la oportunidad de trabajar en el rubro de la seguridad.

A mi Director de Tesis Ing. Juan Alejandro Devincenzi, por aceptar ser mi tutor, estar pendiente de todas mis consultas, por confiar en mi trabajo, gracias por su orientación, su paciencia y motivación durante todo el desarrollo de esta tesis.

## Nómina de Abreviaturas

Las siguientes definiciones corresponden al significado de las siglas presentadas en el trabajo traducidas al español:

CD: Despliegue Continuo.

CI: Integración Continua.

QA: Aseguramiento de la calidad.

VM: Máquina Virtual.

CVE: Vulnerabilidades y exposiciones comunes.

SO: Sistema Operativo.

#### Introducción

Los contenedores docker que soportan los servicios desarrollados deben ser actualizados con frecuencia, actualmente uno de los ataques más comunes hacia los contenedores se debe a las fallas encontradas en las imágenes de los SO por librerías desactualizadas. En este contexto se enmarca la investigación de este trabajo que tiene como meta proporcionar una solución a través de la integración de una tarea de actualización dentro del pipeline de pruebas en Jenkins como una prueba más dentro de una CI.

Este trabajo está dirigido para un servicio web desarrollado en el framework Spring Boot, si bien existe un conjunto de herramientas de desarrollo variadas se tomó Spring Boot debido a la facilidad que brinda la implementación de una aplicación web ya que integra un servidor web incorporado el cual a través de maven "un compilador de código abierto" puede ser empaquetado en un "jar" junto con la aplicación desarrollada, esto facilita el despliegue del servicio en un contenedor Docker.

El objetivo de esta tarea de actualización en el pipeline es la de verificar si el SO de la imagen sobre la que es desplegada el servicio no posee vulnerabilidades y que el servicio web desarrollado en el framework Spring Boot funciona de manera correcta en la versión latest del SO utilizando diferentes herramientas de pruebas funcionales y de seguridad para el despliegue automático y actualizado.

La hipótesis en la que se inscribe este trabajo y que se intentará verificar a lo largo del mismo es. "Se puede integrar pruebas de actualización sobre las imágenes de los SO utilizando las herramientas de Jmeter, Lightning y Trivy para servicios web Spring boot desplegados dentro de contenedores Docker, integrando al equipo de seguridad, QA y operaciones".

Con esta idea el trabajo proporciona una solución en la actualización de la infraestructura de contenedores Docker que exponen servicios web Spring Boot como parte de un flujo de una CI, la tarea de pruebas de actualización se agrega como un componente más detrás de la tarea de pruebas de seguridad y antes de la tarea de despliegue. Para este trabajo se tomó en cuenta la construcción de la tarea de actualización dentro del pipeline que es la solución propuesta, dado que la integración de este esta tarea se ejecuta dentro de un entorno de varias etapas se propone utilizar la metodología "Top-Down" de acuerdo a las herramientas utilizadas y el objetivo detrás de cada uno de ellos.

## 1. Diseño y arquitectura de la seguridad en un ambiente CI/CD

En el caso de la CI/CD existen diversas configuraciones de ambiente de acuerdo a la necesidad de la organización, por lo general se busca tener un ambiente orientado exclusivamente a las pruebas y otro ambiente orientado a producción, para el caso de Jenkins que es un servidor de automatización que posee integraciones automáticas con repositorios como GitHub o Bitbucket es más sencillo enlazar los datos y procesos por medio de repositorios.



Figura 1. Entornos de una CI/CD.

## 1.1 Entorno de Pruebas

En un entorno de pruebas se tiene un repositorio orientado al ambiente de pruebas el cual Jenkins ejecuta una acción en respuesta a algún cambio, el entorno de prueba trabaja con datos aislados clonados del ambiente de producción de forma a emular el entorno productivo.



Figura 2. El Entorno de Pruebas y los datos.

## 1.2 Entorno de Producción

En caso de que se quiera aplicar alguna prueba dentro del ambiente de producción estas deben ser realizadas con mucho cuidado y análisis de forma a que en caso de algún cambio los datos sean reversados a la versión de producción una vez finalizadas las pruebas, se recomienda no realizar pruebas a no ser que sean pruebas unitarias a través de emulaciones sin conexiones a la base de datos.



Figura 3. El Entorno de Producción y los datos.

# 1.3 Entorno de Prueba con Jenkins

Las pruebas de actualización se llevan a cabo en el entorno de pruebas ejecutado previo al entorno de producción.



Figura 4. Entorno de Pruebas.

El entorno de prueba está basado en un pipeline de Jenkins. Un pipeline de Jenkins permite asociar varios trabajos y ejecutarlos en la secuencia que mejor nos convenga, permite aclarar las etapas y los procesos de entrega del software de forma a que se pueda verificar su funcionamiento, evitar errores y obtener una visibilidad de los cambios a través del flujo.

El siguiente pipeline se encuentra en el entorno de pruebas, para este trabajo nos centraremos en la tarea CI\_PruebasActualizacion.



#### Figura 5. Pipeline DevOps [8].

El entorno de pruebas se utiliza exclusivamente para las pruebas de actualización, las pruebas funcionales, las pruebas de seguridad y las pruebas de calidad.

# 2. Diseño y arquitectura del pipeline en la tarea CI\_PruebasActualizacion.

La tarea CI\_PruebasActualizacion tiene dos partes, una que realiza la prueba funcional de la aplicación y otra que realiza una prueba de seguridad sobre los contenedores Docker.

La tarea despliega en primer lugar 2 contenedores Docker con el servicio Empleado, un contenedor en la versión de producción y otro contenedor en la versión latest.

Para las pruebas funcionales se realiza una prueba de estrés sobre ambos servicios desplegados, en caso de algún error se detiene el pipeline.

Las pruebas de seguridad se realizan a través de un análisis de vulnerabilidades sobre el contenedor, cabe destacar que el pipeline se detendrá sólo en caso de una vulnerabilidad con severidad crítica en el contenedor de la versión latest.



Figura 6. Diseño del pipeline de Actualización.

El resultado de este esta tarea dentro del pipeline es un reporte de vulnerabilidades para el equipo de seguridad y un reporte de pruebas de estrés al Equipo de QA, de forma a que en caso de una anomalía en el entorno de pruebas o producción se pueda tener un trazabilidad de las pruebas que fueron aplicadas al desarrollo web.

## 3. Pruebas de actualización en la organización.

Para lograr realizar las pruebas de actualización es necesario integrar a los equipos de Seguridad, QA y Operaciones de forma a que trabajen de manera coordinada, cada equipo debe realizar una tarea específica de acuerdo a su responsabilidad.



Figura 7. Esquema de Pruebas Seguridad, QA, Operaciones y Desarrollo.

El área de Desarrollo es el encargado de construir el servicio y brindar al área de QA las peticiones y respuestas necesarias para la ejecución de la aplicación, el área de QA procesa estas peticiones y genera un entorno de prueba en base al servicio desarrollado.

El equipo de seguridad se encarga de seleccionar las herramientas y establece un criterio de aceptación mínima para la implementación segura del sistema por medio de pruebas de seguridad, el área de Operaciones se encarga de implementar las soluciones propuestas dentro de la CI.

## 3.1 Pruebas de actualización y Desarrollo

Las pruebas de actualización están orientadas a servicios web basados en la plataforma Spring Boot y desplegados en contenedores Docker, es por ello que se toma como base para las pruebas de actualización un servicio web que permite interactuar con los datos del empleado de una organización ficticia.

El servicio Empleado será desarrollado sobre la plataforma JAVA con el JDK en su versión 11.0.11 y como entorno de desarrollo se utilizara la herramienta Apache Netbeans IDE 12.3 todo sobre un SO Linux Ubuntu 20.04.2 LTS.

JAVA: Lenguaje de programación orientado a objetos para desarrollo de aplicaciones.

Apache Netbeans: NetBeans es un entorno de desarrollo integrado libre hecho principalmente para el lenguaje de programación Java.

Linux Ubuntu: Ubuntu es un sistema operativo de software libre y código abierto. Es una distribución de Linux basada en Debian.

Como primer paso debemos instalar el JDK Java para ello lo descargamos desde su página oficial [2].

ORACLE Q Products In	ndustries Resources Support	Events Developer	② View Accounts	Contact Sales
Important Oracle JDK License Upd	late			
The Oracle JDK License has changed for The new Oracle Technology Network License Agreement for personal use and development use, at no cost but other us downloading and using this product. An FAQ is available here	Or releases starting A Oracle Java SE is substantially ses authorized under prior Orac e.	April 16, 2019. different from prior Oracle JDK lice cle JDK licenses may no longer be a	enses. The new license permits certain available. Please review the terms care	n uses, such as efully before
Commercial license and support is available with a low cost 3	Java SE Subscription.	ava not		
oracle also provides the latest openably release ander the o	perisource of Electrise at jukij	uvu.net.		
See also: Java Developer Newsletter: From your Oracle account, select <b>Sub</b>	scriptions, expand Technolog	<b>y</b> , and subscribe to <b>Java</b> .		
Java Developer Day hands-on workshops (free) and other events Java Magazine	5			
JDK 11.0.11 checksum				
Java SE Development Kit 11.0.11 This software is licensed under the Oracle Technology Network L	icense Agreement for Oracle Ja	iva SE		
Product / File Description	File Size		Download	
Linux ARM 64 Debian Package	145.61 MB		'₩ jdk-11.0.11_linux-aarch64	_bin.deb

Figura 8 Página de descarga JAVA JDK 11.

Como en este caso el SO es un Ubuntu descargamos la versión basada en Debian y lo instalamos, en este caso el paquete posee un wizard que nos facilita la instalación.

	jdk-11.0.11
jdk-11.0 Java Plati	<b>D.11</b> form Standard Edition Development Kit
Instalar	
The Java Platform Star the Java platform class tools). The JDK is a dev with the Java Platform	ndard Edition Development Kit (JDK) includes both the runtime environment (Java virtual machine, ses and supporting files) and development tools (compilers, debuggers, tool libraries and other relopment environment for building applications, applets and components that can be deployed Standard Edition Runtime Environment.
Detalles	
Versión	11.0.11-1
Actualizado	Nunca
Licencia	Propietaria
Fuente	jdk-11.0.11_linux-aarch64_bin.deb
Tamaño instalado	152,7 MB
Tamaño descargado	0 bytes

Figura 9 Wizard de instalación JDK 11.0.11.

Una vez instalado la plataforma JDK nos dirigimos a instalar el NetBeans desde su página oficial [3].

Downloading Apache NetBeans 12.3

Apache NetBeans 12.3 was released March 3, 2021. See Apache NetBeans 12.3 Features for a full list of features.

Apache NetBeans 12.3 is available for download from your closest Apache mirror.

- Binaries: netbeans-12.3-bin.zip (SHA-512, PGP ASC)
- Installers:
  - Apache-NetBeans-12.3-bin-windows-x64.exe (SHA-512, PGP ASC)
  - Apache-NetBeans-12.3-bin-linux-x64.sh (SHA-512, PGP ASC)
  - Apache-NetBeans-12.3-bin-macosx.dmg (SHA-512, PGP ASC)

macOS versions prior to 10.14.4 require the Swift 5 Runtime to be installed to launch Apache NetBeans 12.3.

- Source: netbeans-12.3-source.zip (SHA-512, PGP ASC)
- Javadoc for this release is available at https://bits.netbeans.org/12.3/javadoc

Officially, it is important that you verify the integrity of the downloaded files using the PGP signatures (.asc file) or a hash (.sha512 files). The PGP keys used to sign this release are available here.

Apache NetBeans can also be installed as a self-contained snap package on Linux.

## **Deployment Platforms**

Apache NetBeans 12.3 runs on JDK LTS releases 8 and 11, as well as on JDK 15, i.e., the current JDK release at the time of this NetBeans release.

The current JDKs have an issue on macOS Big Sur, that causes freezes on dialogs. That could be fixed by applying the workaround described at NETBEANS-5037.

Figura 10 Página de descarga Apache Netbeans 12.3.

Se selecciona el archivo para la descarga de acuerdo a nuestro SO, en este caso como trabajamos con Ubuntu descargamos la versión para Linux.



Figura 11 Archivo de instalación Apache NetBeans 12.3.

Se descarga un fichero con extensión sh el cual ejecutamos en nuestra máquina para iniciar la instalación.

Descargas> sh Apache-NetBeans-12.3-bin-linux-x64.sh

Figura 12 Comando sh para instalación de Apache NetBeans 12.3.

Deployment Platforms Building from Source Community Approval Earlier Releases Se crea el proyecto web para el servicio Empleado utilizando el Framework Spring Boot, este framework ofrece una página donde se puede personalizar la configuración para luego descargarlo y abrirlo en nuestro entorno de desarrollo [4].

💋 spri	ng initializr							
Project Maven Project Spring Boot 2.6.0 (SNAPS	Lang t Orade Project Ja shOT) O 25.3 (SNAPSHOT) I	uage va O Kotlin O Groovy 2.5.2 O 2.4.9 (SNAPSHOT)		Dependencies No dependency sele	cted		ADD DEPENDENCIES C	TRL + B
O 24.8 Project Metada Group	ata com.example							
Artifact Name	demo							
Description Package name	Com.example.demo							
Packaging Java	● Jar O War O 16 ● 11 O 8							
						]		
		GENERATE CTRL + d	EXPL	DRE CTRL + SPACE	SHARE			

Figura 13 Página de creación para aplicaciones Spring Boot.

Para este trabajo en particular se descargó un servicio de ejemplo de Building REST Services with Spring [5] el cual despliega una serie de clases aplicando el framework spring boot que implementa un servicio de empleados.

## **Building REST Services with Spring**

Table of Contents **Getting Started** The Story so Far... **HTTP** is the Platform What makes something RESTful? Simplifying Link Creation Evolving REST APIs └ Supporting changes to the API **Building links into your REST API** Summarv REST has quickly become the de-facto standard for building web services on the web because they're easy to build and easy to consume. There's a much larger discussion to be had about how REST fits in the world of microservices, but - for this tutorial let's just look at building RESTful services. Why REST? REST embraces the precepts of the web, including its architecture, benefits, and everything else. This is no surprise given its author, Roy Fielding, was involved in probably a dozen specs which govern how the web operates

What benefits? The web and its core protocol, HTTP, provide a stack of features:

• Suitable actions ( GET , POST , PUT , DELETE , ...)

## Figura 14 Building REST Services with Spring.

Abrimos esta aplicación desde nuestro Apache NetBeans.



Figura 15 Proyecto Spring Boot Employee.

Se despliega un conjunto de clases java que crea un servicio web llamado Employee con Spring Boot que posee 4 métodos get, post, put y delete el cual interactúa a través de anotaciones específicas get @GetMapping, post @PostMapping, put @PutMapping, delete @DeleteMapping para las operaciones de Alta, Baja, Modificación y Consulta de Datos. La implementación y los otros métodos asociados a estas anotaciones no se encuentran contemplados en el alcance de este trabajo, para más información de como crear aplicaciones Spring Boot puede dirigirse a [1].



Figura 16 Clase EmployeeController.

Construimos la aplicación con la plataforma de desarrollo Apache Netbeans, para ello nos posicionamos en la pestaña Project seleccionamos el proyecto rest damos click derecho sobre ello y elegimos la opción Clean and Build, esto inicia la construcción de la aplicación.



Figura 17 Resultado de la compilación del proyecto Spring Boot.

Como se observa en la Figura 17 la salida del proyecto es un jar generado llamado rest-0.0.1-SNAPSHOT.jar, que al ejecutarlo de forma directa nos despliega el servicio en un tomcat embebido.

0.0.1-SNAPSHOT> java -ja	ar rest-0.0.1-SNAF	SHOT.jar		
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	,,,, -, -, -, -, -, -, -, -,			
2021-07-19 11:38:01.018 /org/springframework/gu/ 2021-07-19 11:38:01.021	INFO 12981 [ ides/rest/0.0.1-SM INFO 12981 [	main] payroll.Pay APSHOT/rest-0.0.1-SNAPSHOT.j	rollApplication ar started by dani in /home/da	: Starting PayrollApplication v0.0.1-SNAPSHOT on dani-UX310UAK with PID 12981 (/home/dani/.m2/repository ni/.m2/repository/org/springframework/guides/rest/0.0.1-SNAPSHOT) : No active profile set fallion back to default profiles: default
2021-07-19 11:38:02.007				: Bootstrapping Spring Data JPA repositories in DEFERRED mode.
2021-07-19 11:38:02.100				: Finished Spring Data repository scanning in 80ms. Found 1 JPA repository interfaces.
2021-07-19 11:38:02.940				: Tomcat initialized with port(s): 8989 (http)
2021-07-19 11:38:02.954				: Starting service [Tomcat]
2021-07-19 11:38:02.954				: Starting Servlet engine: [Apache Tomcat/9.0.35]
2021-07-19 11:38:03.048				: Initializing Spring embedded WebApplicationContext
2021-07-19 11:38:03.048				: Root WebApplicationContext: initialization completed in 1961 Ms
2021-07-19 11:38:03.281		mathj o.s.s.concu		: Initializing Executo service application askexecutor
2021-07-19 11:38:03.292		maini com zaxxer.		: Mikai Pool-1 - Star Lung
2021-07-19 11:38:03.688				- HHHM0704: Procession Persistence  nitInfo [name: default]
2021-07-19 11:38:03.759				spring in open-in-view is enabled by default. Therefore, database queries may be performed during vie
w rendering. Explicitly	conflaure spring.	ipa.open-in-view to disable	this warning	
2021-07-19 11:38:03.868				: HHH000412: Hibernate ORM core version 5.4.15.Final
2021-07-19 11:38:04.167				: HCANN000001: Hibernate Commons Annotations {5.1.0.Final}
2021-07-19 11:38:04.391				: HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
2021-07-19 11:38:05.245				: HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.N
pJtaPlatform]				
2021-07-19 11:38:05.257				: Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-07-19 11:38:05.826				: Tomcat started on port(s): 8989 (http) with context path '
2021-07-19 11:38:05.827				: Iriggering deterred initialization of Spring Data repositories
2021-07-19 11:38:05.922				: Spring Data repositories initialized!
2021-07-19 11:38:05.937		main] payroll loa		· Stated Payrottappttation in S.355 Seconds (JWH Tuning for 6.192)
2021-07-19 11:38:00.003		maini payrott.Loa		- Preloading Employee[td=1, hake= bitob Baggins , role= bligins ]
1021 01 15 11:38:00.007	100 12901	noterj payrott.toa	doucouse .	. The country improve (come in the baggins , role inter )

Figura 18 Salida del proyecto Spring Boot al iniciar.

← → C ① locahost3999/employees/1
# Aplicaciones @ lgohr O CodigoJavaPr... @ gespro # slack I Remote wrk CL... ∞ VENDO AMPL... S Automatizand... O Copias de seg... @ Hands-On Sec... D Webinar: Trivy... I Atlassian Bitb...
{"id":1, "name":"Bitbo Baggins", "role": "burglar", "\_links"; {"self"; "href": "http://localhost:8989/employees/1"}, "employees"; {"href": "http://localhost:8989/}

## Figura 19 Respuesta del servicio.

El objetivo será que este servicio pase por la tarea de prueba de actualización dentro del pipeline en la CI por medio de la coordinación de los equipos de Seguridad, QA, Operaciones y Desarrollo.

## 3.2 Pruebas de actualización y QA

Para las pruebas de actualización se debe contar con un entorno de pruebas de estrés por cada servicio que se pretende desplegar elaborado por el departamento de QA.



Figura 20 Generación de Entorno de Prueba QA.

El equipo de desarrollo se encarga de brindar las especificaciones del servicio al equipo de QA y este último elabora un entorno de pruebas acorde a lo especificado por el equipo de desarrollo.

Este es un paso previo a toda prueba de actualización que se desea realizar, para este fin existen diversas herramientas como el caso de JMeter que es un proyecto de Apache que puede ser utilizado como una herramienta de prueba de carga para analizar y medir el rendimiento de una variedad de servicios [11].

El Apache Jmeter puede ser descargado desde su página oficial [15] e instalado de acuerdo al SO que posea, en este caso particular se utiliza para un SO Ubuntu 20.04.2 LTS, un requisito previo para el funcionamiento de esta herramienta es que es necesario tener previamente instalado Java en su versión 1.8 para arriba.

#### Apache JMeter 5.4.1 (Requires Java 8+)

#### Binaries

apache-jmeter-5.4.1.tgz sha512 pgp apache-jmeter-5.4.1.zip sha512 pgp

#### Source

apache-jmeter-5.4.1\_src.tgz sha512 pgp apache-jmeter-5.4.1\_src.zip sha512 pgp

#### Archives

Older releases can be obtained from the archives.

browse download area
Apache JMeter archives...
Apache Jakarta JMeter archives...

Figura 21. Página de descarga de Apache Jmeter.

Se extrae el paquete que contiene el programa, se accede a la sub carpeta de apache jmeter "bin" y se ejecuta el script jmeter.sh esto iniciará la plataforma del programa.

<pre>&gt;&gt; cd apache-jmeter-5.3/ apache-jmeter-5.3&gt; cd bin bin&gt; ls ponterenter see BeanShellAssertion.bshrc BeanShellAssertion.bshrc BeanShellAssertion.bshrc create-rmi-keystore.bshrc create-rmi-keystore.sh bin&gt; sudo sh jmeter.sh</pre>	examples hc.parameters heapdump.cmd heapdump.sh jaas.conf jmeter jmeter.bat	jmeter.log jmeter.n.cmd jmeter.properties jmeter.server jmeter.server.bat jmeter.sh	jmeter-t.cmd jmeterw.cmd krb5.conf log4j2.xml mirror-server mirror-server.cmd mirror-server.sh	reportgenerator.properties report-template saveservice.properties shutdown.sh stoptest.cmd stoptest.sh	system.properties templates testResult.jtl threaddump.cmd upgrade.properties user.properties	utility.groovy		
			Apache JMet	ter (5.3)				
Arthon Ediar Seeth Larace Occores Tool for a grant Larace Accores Tool for a grant Larace Accores Tool A Plan de Proebse	- Ayuda - Ay	Pian de Pruebas Nombre: Plan de Pruebas Comentarios	3 Nombre:				<u>A</u> 0	a/o 🕀

Figura 22 Apache Jmeter.

Apache Jmeter posee una serie de opciones que nos permite emular peticiones web con características específicas de acuerdo a la necesidad que podamos tener, en este caso el equipo de QA debe crear un entorno de prueba de acuerdo a las características del servicio.

Se crea un entorno de pruebas de estrés para el servicio desarrollado, luego se agrega un grupo de hilos para configurar la ejecución de las pruebas desde el IDE de Apache Jmeter para ello damos click derecho sobre el plan que generamos, seleccionamos Añadir Hilos/Grupo de hilos..

Dian de Dra	ushao.			
	Añadir	Þ	Hilos (Usuarios)	Grupo de Hilos
	Pegar	Ctrl-V	Elemento de Configuración 🕨	Montar grupo de Hilos

Figura 23 Apache Jmeter opción configurar hilos.

El grupo de hilos indica a JMeter que tendrá que lanzar un número de peticiones y como se comportan estas peticiones. Lo que se pretende es simular el funcionamiento de un usuario o cliente web que realiza peticiones a un servidor web.

<ul> <li>▲ Plan de Pruebas</li> <li>◆ Usuarios</li> <li>★ Gestor de Cabecera HTTP</li> <li>▶ ✓ Get Empleado</li> <li>▶ ✓ Post Empleado</li> </ul>	Grupo de Hilos Nombre: Usuarios Comentarios 1				
<ul> <li>/ Put Empleados</li> <li>/ Delete Empleado</li> </ul>	Acción a tomar después de un error de Muestreador                • Continuar             • Comenzar siguiente iteración             • Parar Hilo             • Parar Test             • Parar test ahora             2				
	Número de Hilos     10       Periodo de Subida (en segundos):     2       Contador del bucle:     Sin fin	3			

Figura 24 Opción de Hilos Jmeter.

En el punto 1 podemos agregar un nombre para nuestro grupo de hilos así como un comentario descriptivo, en el paso 2 se enumera una serie de acciones en caso de existir un error, para este caso en particular queremos que se continúe ya que se pretende utilizar el reporte de salida del Jmeter dentro de la CI para avanzar o no en el flujo, en el punto 3 se configura las propiedades del hilo.

Número de hilo: Indica cuántos usuarios simultáneos realizan la petición al servicio, en este caso definimos 10 hilos.

Periodo de subida en segundos: Indica el intervalo de tiempo que se espera entre cada petición, en este caso tenemos 2 segundos entre cada petición.

Contador del bucle: Tiene 2 opciones, sin fin para que la prueba no finalice hasta que se lo indiquemos y la otra donde pasamos una parámetro para definir la cantidad de veces a probar, en este caso definimos 10 peticiones por cada acción HTTP.

Se crean las peticiones HTTP dentro del grupo de hilos que se creó anteriormente para cada uno de los métodos del servicio el get, post, put y delete, para agregar estas peticiones se ingresa al Ide de Apache Jmeter se posiciona sobre el grupo de hilos damos click derecho Añadir/Muestreador/Petición HTTP.



Figura 25 Configuración de una petición HTTP en Apache Jmeter.

En la pantalla se despliega una serie de opciones que nos permite personalizar una petición, en la parte 1 podemos definir el nombre de la petición HTTP y un comentario descriptivo, en la parte 2 se define el protocolo HTTP o HTTPS en caso de no tener ningún valor se toma HTTP por defecto, definimos la ubicación del servidor en este caso localhost ya que el servicio se ejecuta de forma local y el puerto donde está publicado el servicio 8585, en la parte 3 se define el método HTTP Get, Post, Put, Delete, Head, Options, Trace, Patch y la ruta donde se encuentra publicado el servicio, en la parte 4 se definen los parámetros que se le pasa a la petición.

Como el servicio posee solo 4 peticiones Get, Post, Put, Delete se debe crear una petición HTTP por cada método.



Figura 26 Peticiones Http Get, Post, Put, Delete en Apache Jmeter.

A Find the multiple	Petición HTTP Nombre: Gel Engleado Comentanos  Service Serviced Serviced Petición HTTP Petición HTTP (artición HTTP)	Petición HTTP Nembre - (set Empleado Comentarios Sanada Amond Sanada Sanada Sanada Petición ITP				
					Jncluir Equals?	

## Figura 27 Petición Get.

A Plan de Privebas     Veloro de Colacera HTTP     Veloro de Colacera HTTP     Veloro de Enpleado     Prut Empleado     Velete Empleado	Petidión HTTP Itemère Petit Empleado Conerctairos  Secied Advanced  Petidión Inter Petidión Inter Petidión Inter Petidión Inter Potor  Petidión Inter Petidión I	Pueto (8585
	Presenters General State (1997) (1997	

Figura 28 Petición Post.

<ul> <li>▲ Plan de Pruebas</li> <li>♥ Usuarios</li> <li>♦ Gestor de Cabecera HTTP</li> <li>&gt; ✓ Get Empleada</li> <li>&gt; ✓ Post Empleada</li> </ul>	Petición HTTP Nombre: [Put Empleados Conentarios ]	
P      Celete Empleado      Colete Empleado	**************************************	
	PUT              • Rutor / temployees/0            Pedrop Automiktorinete         Seyle Redectiones         Ublas Krepelive         Usar trudpactform-dat para HTT POST         Calecoras compatibles on naingadores           Parameters         Body Dista         Files Upload         I         Calecoras (Trudpact)           Image: 1/Jostbrer', role*: Polition*()                  endop() Statismeter (Calecoras Compatibles on naingadores	Codificacion del contenido:

## Figura 29 Petición Put.

- 1 -	Flan de Pruebas	Petición HT Nombre: Comentarios	TP Delete Empleado				11				
	> 🖋 Delete Empleado	Basic Advanced									
										nido:	
E'											ancluir B

Figura 30 Petición Delete.

El siguiente paso al tener construido todo el entorno de pruebas es configurar la forma en que deseamos visualizar los resultados, Apache Jmeter posee un conjunto de opciones llamados Listener encargados de mostrar los resultados de las muestras en una serie de formatos establecidos, en este caso particular optamos por la opción Árbol de Resultados que nos permite ver tanto la solicitud como la respuesta de una petición HTTP, para configurar este listener nos dirigimos a la petición HTTP y damos click derecho, seleccionamos la opción Añadir/Receptor/Ver Àrbol de Resultados.



Figura 31 Resultado de las pruebas en Jmeter.

Por último se guarda el proyecto con el nombre Empleado.jmx, de forma que más adelante pueda ser utilizado en el pipeline dentro de la tarea de actualización.



Figura 32 Proyecto Empleado.jmx.

#### 3.3 Pruebas de actualización y Seguridad

Hasta este momento se ha descrito la interacción del departamento de desarrollo y QA para las pruebas con Apache Jmeter, en esta sección se describen las tareas necesarias que debe llevar el departamento de seguridad para integrar las pruebas de actualización en el flujo de la CI.

Las pruebas de actualización incluyen dentro de si pruebas funcionales de la aplicación y análisis de vulnerabilidades de la plataforma Docker sobre el cual es desplegado el servicio, la configuración de las pruebas funcionales se describen en el capítulo 3.2 Pruebas de actualización y QA, este capítulo se enfoca en el otro factor necesario para la migración a una plataforma más actualizada que es el análisis de seguridad de los contenedores Docker.

En este punto se determina qué condiciones de seguridad se deben cumplir en la tarea de actualización dentro del pipeline en el flujo de la CI, se buscan los siguientes resultados:

El contenedor en la versión latest no debe poseer vulnerabilidades críticas, en caso de que lo tenga el flujo de la CI será detenido..

Un reporte de vulnerabilidades del contenedor en la versión de latest una vez finalizado el pipeline de actualización.

El equipo de seguridad debe seleccionar la herramienta adecuada a la infraestructura de acuerdo a los objetivos esperados, para este fin se optó en este trabajo por la herramienta Trivy que permite realizar un análisis de vulnerabilidades sobre contenedores Docker, pero antes de hablar de Trivy debemos profundizar un poco más sobre la herramienta Docker.

#### 3.3.1 Docker

Para este trabajo se tomó la herramienta Docker para contenedorizar el servicio web Empleado desarrollado en Spring Boot, Docker es una herramienta ideal que automatiza el despliegue de aplicaciones dentro de contenedores en este caso el servicio Empleado, permite desplegar una plataforma de manera personalizada y reducida a través de un SO emulado ,su facilidad para implementarlo lo hace una herramienta ideal para entornos productivos.

Algunas ventajas de Docker:

Es un proyecto de código abierto con una gran comunidad que aporta mucha información.

Puede ejecutar varios contenedores en la misma máquina y compartir el kernel del sistema operativo con otros contenedores cada uno ejecutándose como procesos aislados en el espacio del usuario.

Los contenedores ocupan menos espacio que las VM, las imágenes de los contenedores suelen tener un tamaño de decenas de MB [6].

Para la instalación de Docker se debe seguir la siguiente referencia [7] e instalarlo de acuerdo al SO que se disponga.

# Get Docker

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

You can download and install Docker on multiple platforms. Refer to the following section and choose the best installation path for you.



A native application using the macOS sandbox security model which delivers all Docker tools to your Mac.



Docker Desktop for Windows

A native Windows application which delivers all Docker tools to your Windows computer.



Install Docker on a computer which already has a Linux distribution installed.

Figura 33 Página de Instalación de Docker.

Una vez instalado en la consola se puede ejecutar el comando docker --version para ver la versión instalada.



Figura 34 Versión de Docker instalada.

## 3.3.1.1 Docker File

Si observamos la arquitectura de la tarea de actualización dentro del pipeline en la figura 35 se necesita desplegar el servicio Empleado en dos contenedores Docker, una en la versión Alpine 3.10 y otra en la versión Alpine latest y aplicar pruebas funcionales y análisis de vulnerabilidades sobre ellos, para este fin nos valemos de una característica que posee Docker que es el Dockerfile.



Figura 35 Arquitectura del Dockerfile para el pipeline de actualización.

Docker puede crear imágenes automáticamente leyendo las instrucciones de un archivo Dockerfile, el Dockerfile es un documento de texto que contiene todos los comandos para ensamblar una imagen personalizada y a través del docker build compilarlo de forma automática de manera a ejecutar varias instrucciones de línea de comandos en sucesión [10].

Algunos comandos que se usan en el Dockerfile:

FROM: La instrucción FROM especifica la imagen principal a partir de

la cual se está construyendo.

**RUN:** La instrucción RUN ejecuta cualquier comando en una nueva capa encima de la imagen actual, la imagen resultante se utilizará para el siguiente paso en el Dockerfile.

**ENV:** Establece el valor de una variable de entorno, el valor estará disponible para todas las etapas posteriores en el proceso de compilación de la imagen.

**COPY:** La instrucción COPY copia archivos o directorios del servidor y los agrega al sistema de archivos del contenedor en la ruta especificada como destino.

**WORKDIR:** La instrucción WORKDIR establece el directorio de trabajo para cualquier comando como RUN, CMD, ENTRYPOINT, COPY, etc y las instrucciones que le siguen en el Dockerfile.

**CMD:** El propósito principal del comando CMD es proporcionar valores predeterminados para un contenedor en ejecución.

Como se mencionó en la tarea de actualización dentro del pipeline se tiene dos versiones de Dockerfile, una levanta el servicio web desarrollado en el SO de la versión de producción en este caso alpine:3.10 y otro Dockerfile que despliega el mismo servicio en la versión alpine:latest, pero antes de hablar del flujo del Dockerfile se debe mencionar el SO sobre el cual será levantado el servicio Empleado que es el Apline.

#### Alpine

Alpine Linux es una distribución de Linux ligera y orientada a la seguridad basada en musl libc y busybox, es un SO Linux reducido que no requiere más de 8 Mb de espacio en comparación a los 130 MB que necesita un SO como mínimo para instalar, este SO fue elegido para el trabajo debido a su seguridad ya que todos los binarios se compilan como ejecutables independientes de la posición (PIE) lo que permite la ejecución de los programas en secciones aleatorias en la memoria [16].

En este trabajo se tiene el servicio empleado dentro de un contenedor Docker basado en una imagen Alpine 3.10 y se lo pretende actualizar a un Alpine:latest.

#### Flujo del Dockerfile

Como primer paso se instala el SO en este caso el alpine, luego se instala el jdk que corresponde para levantar la aplicación en este caso el jdk11, se define la variable de entorno JAVA\_HOME, se copia el JAR del servicio Empleado del directorio actual al directorio interno del contenedor /usr/src/servicio, se posiciona sobre este directorio y se ejecuta el JAR que levanta la aplicación, cabe mencionar que el JAR rest-0.0.1-SNAPSHOT.jar debe estar en el mismo directorio donde se encuentra el Dockerfile.

```
FROM alpine:latest
```

# Se installa el OpenJDK-11.
RUN apk add openjdk11

# Variable de Entorno JAVA ENV JAVA\_HOME="/usr/lib/jvm/default-jvm/"

# Has to be set explicitly to find binaries
ENV PATH=\$PATH:\${JAVA\_HOME}/bin

COPY . /usr/src/servicio

WORKDIR /usr/src/servicio

CMD [ "java", "-jar", "rest-0.0.1-SNAPSHOT.jar" ]

Figura 36 Dockerfile con SO alpine en su versión latest.
```
FROM alpine:3.10
```

```
# Se installa el OpenJDK-11.
RUN apk add openjdk11
```

```
# Variable de Entorno JAVA
ENV JAVA_HOME="/usr/lib/jvm/default-jvm/"
```

```
# Has to be set explicitly to find binaries
ENV PATH=$PATH:${JAVA_HOME}/bin
```

COPY . /usr/src/servicio

WORKDIR /usr/src/servicio

```
CMD [ "java", "-jar", "rest-0.0.1-SNAPSHOT.jar" ]
```

Figura 37 Dockerfile con SO alpine en su versión producción alpine 3.10.

Para construir el contenedor con la imagen a partir del Dockerfile se utiliza el Docker Build, el comando docker build crea imágenes de Docker a partir de un Dockerfile y un contexto, para ejecutar este comando se escribe el comando docker build [path], se debe indicar el directorio donde está el Dockerfile y se corre el comando, esta acción inicia la construcción del contenedor con la imagen de acuerdo a la configuración del Dockerfile.

```
app1> docker build .
Sending build context to Docker daemon 38.89MB
Step 1/7 : FROM alpine:3.10
.10: Pulling from library/alpine
396c31837116: Pull complete
Digest: sha256:451eee8bedcb2f029756dc3e9d73bab0e7943c1ac55cff3a4861c52a0fdd3e98
Status: Downloaded newer image for alpine:3.10
    -> e7b300aee9f9
Step 2/7 : RUN apk add openjdk11
 ---> Running in 028d3f3ba914
fetch http://dl-cdn.alpinelinux.org/alpine/v3.10/main/x86_64/APKINDEX.tar.gz
fetch http://dl-cdn.alpinelinux.org/alpine/v3.10/community/x86_64/APKINDEX.tar.gz
1/30) Installing openjdk11-jmods (11.0.4_p4-r1)
2/30) Installing openjdk11-demos (11.0.4_p4-r1)
3/30) Installing openjdk11-doc (11.0.4_p4-r1)
4/30) Installing java-common (0.2-r0)
5/30) Installing libffi (3.2.1-r6)
6/30) Installing p11-kit (0.23.16.1-r1)
7/30) Installing libtasn1 (4.14-r0)
(8/30) Installing p11-kit-trust (0.23.16.1-r1)
(9/30) Installing ca-certificates (20191127-r2)
10/30) Installing java-cacerts (1.0-r0)
11/30) Installing openjdk11-jre-headless (11.0.4_p4-r1)
12/30) Installing libxau (1.0.9-r0)
13/30) Installing libbsd (0.9.1-r1)
 14/30) Installing libxdmcp (1.1.3-r0)
 15/30) Installing libxcb (1.13.1-r0)
```

Figura 38 Construcción del Dockerfile con SO alpine en su versión producción alpine 3.10.

### 3.3.1.2 Docker Compose

Para simplificar aún más las cosas en el despliegue del servicio Empleado en ambos contenedores se puede utilizar el Docker Compose.

Docker Compose es una herramienta de Docker que se utiliza para definir y ejecutar aplicaciones de varios contenedores a través de un archivo en formato YAML de manera a iniciar todos los servicios desde su configuración, esta es una manera muy eficiente y rápida de iniciar aplicaciones en los contenedores, como en este trabajo la tarea de actualización del pipeline se basa en probar un mismo servicio en dos plataformas, uno en una versión de producción y otra en una versión actualizada nos conviene utilizar este medio, configuramos un archivo en formato YAML e iniciamos ambas aplicaciones al mismo tiempo con un solo comando, esto facilita la CI.



Figura 39 Esquema para archivo Docker Compose.

Para la utilización del Docker Compose necesitamos organizar los Dockerfile en ficheros de manera que al configurar el archivo YAML se pueda configurar la ubicación de los Dockerfile.

Para este fin necesitamos organizar los archivos, en este caso creamos una carpeta de servicios y dentro de ella tenemos dos carpetas una para el Dockerfile que levanta el servicio en el SO en la versión latest app1 y otra para el Dockerfile que levanta el servicio en el SO en la versión de producción app2 y dentro de la carpeta Servicios el archivo de docker-compose.



Figura 40 Organización de ficheros para el docker-compose.

Esta organización nos permite configurar el archivo docker compose de la siguiente forma.

```
1 version: '3'
2
    services:
 3
      app1:
4
        build: ./app1
 5
         networks:
 6
         redpr:
 7
             aliases:
 8
               - app1
9
        ports:
           - "8989:8989"
10
11
       app2:
         build: ./app2
12
13
        networks:
14
           redpr:
             aliases:
15
16
               - app2
17
         ports:
           - "8686:8989"
18
19
    networks:
20
       redpr:
```

Figura 41 Archivo docker-compose.

Este archivo posee diferentes tags.

version: Indica la versión del docker-compose.

**services**: El tag services indica que se va a configurar un servicio, en este caso tenemos el app1 y el app2.

**build**: El tag build indica la ruta donde está el Dockerfile a ejecutar para levantar el servicio.

networks: El tag networks define una red.

**aliases**: Define un alias de conexión para el contenedor dentro de la red creada.

ports: El puerto donde será publicada la aplicación.

Se posiciona sobre la carpeta servicio y se corre el comando docker-compose up -d, esto inicia la construcción de los contenedores con los servicios expuestos de acuerdo al archivo docker-compose.

```
Servicios> docker-compose up -d
Building app1
Step 1/7 : FROM alpine:3.10
---> e7b300aee9f9
Step 2/7 : RUN apk add openjdk11
 ---> Using cache
 ---> fb26fba87460
Step 3/7 : ENV JAVA_HOME="/usr/lib/jvm/default-jvm/"
 ---> Using cache
 ---> bceb12dda77d
Step 4/7 : ENV PATH=$PATH:${JAVA_HOME}/bin
 ---> Using cache
 ---> 36f6ae10de14
Step 5/7 : COPY . /usr/src/servicio
---> 3d82f2389682
Step 6/7 : WORKDIR /usr/src/servicio
 ---> Running in c3a8fc8379fe
Removing intermediate container c3a8fc8379fe
 ---> 41d33d7925df
Step 7/7 : CMD [ "java", "-jar", "rest-0.0.1-SNAPSHOT.jar" ]
```

Figura 42 Salida de consola una vez ejecutado el docker-compose.yaml.

Si se ejecuta el comando docker ps, se puede ver que los contenedores están levantados.

Servicios> docker ps CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES 15a2dbc48cc4 servicios\_app1 "java -jar rest-0.0..." 21 seconds ago Up 19 seconds 0.0.0.0:8686->8686/tcp servicios\_app1\_1 ffc1785939f8 servicios\_app2 "java -jar rest-0.0..." 21 seconds ago Up 19 seconds 0.0.0.0:8989->8989/tcp servicios\_app2\_1

Figura 43 Contenedores en ejecución.

### 3.3.2 Trivy

Para la identificación de las posibles vulnerabilidades en los contenedores se utiliza la herramienta Trivy que es un escáner de vulnerabilidades simple y completo para contenedores y otros artefactos. Trivy detecta vulnerabilidades de paquetes de SO Alpine, RHEL, CentOS, etc. con sus dependencias [8].

Vulnerability	Vulnerability	Misconfiguration	Vulnerability	Misconfiguration			
Container Images	Files	ystem	Git Repositories				
Trivy							
Vi	Inerability/Mi	sconfiguration:	Scanner				

Figura 44. Esquema de funcionamiento de la herramienta Trivy.

Para instalar Trivy se ingresa a la página de instalación [9], los instaladores están clasificados de acuerdo a los diferentes SO, en este trabajo se instala para Debian/Ubuntu que es el SO sobre el cual se desea desplegar la tarea de actualización.

### Debian/Ubuntu

Repository DEB

Add repository setting to /etc/apt/sources.list.d.

```
sudo apt-get install wget apt-transport-https gnupg lsb-release
wget -q0 - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key
echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | su
sudo apt-get update
sudo apt-get install trivy
```

Figura 45 Comandos de instalación de Trivy.

Para verificar si la instalación fue exitosa se puede ingresar en la consola el comando trivy --version, si el programa fue instalado exitosamente se desplegará su número de versión.

```
~> trivy --version
Version: 0.18.1
Vulnerability DB:
   Type: Light
   Version: 1
   UpdatedAt: 2020-11-26 12:16:21.369213294 +0000 UTC
   NextUpdate: 2020-11-27 00:16:21.369212494 +0000 UTC
   DownloadedAt: 0001-01-01 00:00:00 +0000 UTC
~>
```

Figura 46 Versión del Trivy instalada.

Trivy posee el comando image para escanear una imagen de un contenedor.

\$ trivy image [YOUR\_IMAGE\_NAME]

Figura 47 Comando para analizar desde una imagen con Trivy.

Se observa en la Figura 43 que existen dos imágenes corriendo en dos contenedores, uno llamado servicios\_app1 y el otro llamado servicios\_app2 que fueron creado a partir del Dockerfile de las figuras 35 y

36, para que Trivy analice una de estas imágenes se utiliza la estructura del comando expuesto en la figura 47, esto inicia el proceso de escaneo en busca de vulnerabilidades.

Servicios> trivy serv 2021-07-26T04:20:09.2 2021-07-26T04:20:09.2 2021-07-26T04:20:09.2 2021-07-26T04:20:09.2 servicios_app1 (alpir	icios_app1 .74-0400 INFO .75-0400 INFO .81-0400 WARN .81-0400 WARN me 3.10.9)	Detecting Alpine vulnerabil Detecting jar vulnerabiliti This OS version is no longe The vulnerability detection	lities ies er supported n may be insu	by the distribution: fficient because sec	alpine 3.10.9 urity updates are not provided	
Total·A (UNKNOWN·A						
10181. 0 (ONNIONIA. 0,	LOW. 0, HEDION. 0,	, HIGH. 0, CRITICAL. 0)				
usr/src/servicio/rest	-0.0.1-SNAPSHOT.ja	r -				
======================================		= 3, HIGH: 6, CRITICAL: 1)				
·						
	LIBRARY	VULNERABILITY I	D   SEVERITY	INSTALLED VERSION	FIXED VERSION	I т
+   net.minidev:json-sm b+	hart	CVE-2021-27568	CRITICAL	2.3	2.4.1, 1.3.2	json-smart: uncaug
						exception may lead
to crash   						or information dis
closure						
 /nvd/cve-2021-27568						>avu.aquasec.com

Figura 48 Salida del comando trivy servicios\_app1.

Trivy permite filtrar las vulnerabilidades por severidad a través de la opción --severity.

## By Severity

Use --severity option.

\$ trivy image --severity HIGH,CRITICAL ruby:2.4.0

Figura 49 Opción severity Trivy.

El equipo de seguridad definió que solo las fallas críticas en el contenedor de la versión latest sea un factor de detención del pipeline, es por ello que podemos filtrar la búsqueda de vulnerabilidades solo por severidad crítica.

Servicios> trivyseverity CRITICAL servicios_app1 2021-07-26T04:57:53.629-0400 INFO Detecting Alpine vulnerabilities 2021-07-26T04:57:53.635-0400 INFO Detecting jar vulnerabilities 2021-07-26T04:57:53.635-0400 WARN This OS version is no longer supported by the distribution: alpine 3.10.9 2021-07-26T04:57:53.635-0400 WARN The vulnerability detection may be insufficient because security updates are not provided							
servicios_app1 (alpine 3.2	10.9)						
Total: 0 (CRITICAL: 0)							
usr/src/servicio/rest-0.0.1-SNAPSHOT.jar ====================================							
+   LIBRARY	VULNERABILITY ID	+ SEVERITY	INSTALLED VERSION	FIXED VERSION	TITLE		
<pre>+ + + + + + + + + + + + + + + + + + +</pre>							
servicios>							

Figura 50 Opción severity CRITICAL Trivy.

Con esta herramientas el equipo de seguridad define el nivel de vulnerabilidad aceptable que puede tener la infraestructura Docker para el servicio Empleado, como el análisis se realiza dentro de una CI necesitamos de alguna manera detener o seguir el flujo dentro del pipeline fallando el flujo de trabajo si se encuentra una vulnerabilidad, para esta situación Trivy posee la opción --exit-code 0 para seguir con la siguiente tarea y --exit-code 1 para fallar y no seguir con la siguiente tarea dentro del flujo del pipeline.

Si le sumamos que solo las vulnerabilidades críticas serán tomadas en cuenta para la detención del pipeline se debe configurar en la herramienta Trivy con la opción --severity CRITICAL, el comando quedaría de la siguiente manera:

trivy image --exit-code 0 --severity CRITICAL [Nombre de la Imagen]

Con esta solución el equipo de seguridad puede cumplir con sus objetivos adecuado a una CI.

### 3.4 Pruebas de Actualización y Operaciones

Una vez que se definió como realizar las pruebas funcionales y las pruebas de seguridad el equipo de operaciones debe encargarse de integrar estos procesos y orquestar de acuerdo al objetivo esperado, como se mencionó con anterioridad la herramienta a utilizar para la CI es Jenkins.

### 3.4.1 Jenkins

Jenkins es un servidor de código abierto escrito en Java, se utiliza para la automatización de entregas en el proceso de desarrollo de software mediante la integración continua y facilita ciertos aspectos de la entrega continua [11], el instalador de Jenkins se encuentra en su página web oficial [12] clasificados de acuerdo a los diferentes SO, en este caso el trabajo utiliza Ubuntu como SO.

#### **Downloading Jenkins**

Jenkins is distributed as WAR files, native packages, installers, and Docker images. Follow these installation steps:

- 1. Before downloading, please take a moment to review the Hardware and Software requirements section of the User Handbook.
- 2. Select one of the packages below and follow the download instructions.
- 3. Once a Jenkins package has been downloaded, proceed to the Installing Jenkins section of the User Handbook
- 4. You may also want to verify the package you downloaded. Learn more about verifying Jenkins downloads.

#### Download Jenkins 2.289.3 LTS for:

Generic Java package (.war) SHA-256: 996dtd29d51933546at9e9t77c29b371tb0627b8266b6c9t134ac2e0t1248b87	Generic Java package (.war) SHA-256: 917batb1981089571td7014036101edace377c7a511dffe393247c2373b0638a
Docker	Docker
Ubuntu/Debian	Ubuntu/Debian
CentOS/Fedora/Red Hat	CentOS/Fedora/Red Hat
Windows	Windows
openSUSE	openSUSE
FreeBSD 🎕	Arch Linux 🚳
Gentoo 🕸	FreeBSD 🎄
macOS 🎕	Gentoo 🎄
OpenBSD 🎕	macOS 🎄
	OpenBSD 🎄
	OpenIndiana Hipster 🚳

Download Jenkins 2.304 for:

Figura 51 Página de Instalación de Jenkins.

Se ejecutan los comandos indicados en la página de instalación dentro de la terminal del SO.

### Jenkins Debian Packages

This is the Debian package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system:

WARNING: The gpg key use to sign our packages has been updated on 16th of April 2020, therefore you need to reimport it if you imported before this date.
wget -q -0 - https://pkg.jenkins.io/debian-stable/jenkins.io.key   sudo apt-key add -
Then add the following entry in your /etc/apt/sources.list:
deb https://pkg.jenkins.io/debian-stable binary/
Update your local package index, then finally install Jenkins:
sudo apt-get update sudo apt-get install jenkins
The apt packages were signed using this key:
pub       rsa4096 2020-03-30 [SC] [expires: 2023-03-30]         62A9756BFD780C377CF24BA8FCEF32E745F2C3D5         uid       Jenkins Project         sub       rsa4096 2020-03-30 [E] [expires: 2023-03-30]

Figura 52 Comandos de instalación de Jenkins.

Al finalizar la instalación se inicia el servicio del servidor Jenkins con el siguiente comando.

# sudo systemctl start jenkins

Figura 53 Comando para iniciar Jenkins.

El servidor se ejecuta en el puerto 8080 de la máquina, para acceder al mismo ejecutamos la url <u>http://localhost:8080</u> e ingresamos la contraseña expuesto en el path que indica la pantalla Figura 54.



Figura 54 Wizard de configuración de Jenkins desbloqueo.

Se escogen los plugins a instalar que necesitamos, en este caso instalamos lo sugerido.



Figura 55 Wizard de configuración de Jenkins selección de plugin.

Se instalan los plugins seleccionados.

Getting Started				
Get	ting Star	ted		
✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	OWASP Markup Formatter ** Oracle Java SE Development Kit
✓ Timestamper	Workspace Cleanup	🗘 Ant	🗘 Gradle	** Structs ** Pipeline: Step API ** Token Macro
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	🗘 Pipeline: Stage View	Build Timeout ** Credentials ** Plain Credentials
🤣 Git	Subversion	SSH Build Agents	O Matrix Authorization Strategy	** SSH Credentials Credentials Binding ** SCM API
PAM Authentication	C LDAP	C Email Extension	Mailer	<pre>** Pipeline: API Timestamper ** Script Security ** Pipeline: Supporting APIs ** Durable Task ** Pipeline: Nodes and Processes ** Juhi ** Command Agent Launcher ** bouncycastle API ** - required dependency</pre>
Jenkins 2.222.4				required dependency

Figura 56 Wizard de configuración de Jenkins instalación de plugin.

Al finalizar este proceso se crea una cuenta para administrar el programa.

Getting Started	
Create First Adm	in User
Username:	
Password:	
Confirm password:	
Full name:	
E-mail address:	
Jenkins 2 222.4	Continue as admin Save and Continue

Figura 57 Wizard de configuración de Jenkins creación de la cuenta.

Al completar todos estos pasos se ingresa a la url http://localhost:8080 y se despliega la página de inicio de Jenkins.

🍓 Jenkins		Q search	🛕 monitor 1	💄 Geekfkare 🛛 🕁 log out
Jenkins →				enable auto refresh
<ul> <li>New Item</li> <li>People</li> <li>Build History</li> <li>Manage Jenkins</li> <li>Ny Views</li> </ul>	Welcome to Jenkins! Please <u>create new jobs</u> to get started.			⊘add.description
New View				
Build Queue -				
Build Executor Status -				

Figura 58 Página de inicio Jenkins.

### 3.4.1.1 Creacion de Tareas Jenkins

Este trabajo está centrado en la configuración de la tarea CI\_PruebasActualizacion, las demás tareas como CI\_Construccion, CI\_PruebaDeCodigo, CI\_PruebaSeguridad y CI\_PruebaDespliegue están solo a modo de ejemplos para establecer el contexto donde se ejecuta el la tarea CI\_PruebasActualizacion dentro del pipeline.

Jenkins trabaja por medio de tareas que configuran el funcionamiento de cada evento, en este caso debemos crear una tarea para ejecutar las pruebas de actualización, a continuación se describen los pasos necesarios para crear una tarea en Jenkins.

Al ingresar en la url <u>http://localhost:8080</u> el navegador despliega la pantalla de inicio de Jenkins a la izquierda de la misma se encuentra una serie de opciones agrupados bajo la etiqueta Panel de Control, para crear una nueva tarea se debe seleccionar la opción Nueva Tarea.

襣 Jenkins
Panel de Control 🕠
🚔 Nueva Tarea
•
🍓 Personas
📄 Historial de trabajos
Relacion entre proyectos
Comprobar firma de archivos
🐡 Administrar Jenkins

Figura 59 Opción Nueva Tarea.

Esta acción despliega una serie de opciones para configurar el tipo de trabajo que queremos crear y un cuadro de texto donde podemos asignar un nombre al trabajo, en este caso seleccionamos la opción de "Crear un proyecto con estilo libre".



Figura 60 Creación de un trabajo en Jenkins.

$\odot$	IÔI	CI_Construccion	2 Mes 1 día - <b>#21</b>	N/D	1 Min 10 Seg	ø
$\odot$	τ <b>ộ</b> χ	CI_PruebasActualizacion	2 Mes 1 día - <b>#50</b>	2 Mes 16 días - <b>#43</b>	3 Min 1 Seg	ø
$\odot$	ΙÔΙ	CI_PruebasDeCodigo	2 Mes 1 día - <b>#39</b>	2 Mes 7 días - <b>#34</b>	1 Min 42 Seg	ø
$\odot$	Ŕ	CI_PruebasDespliegue	2 Mes 1 día - <b>#58</b>	2 Mes 7 días - <b>#54</b>	25 Ms	ø
$\odot$	Ŕ	CI_PruebasSeguridad	2 Mes 1 día - <b>#99</b>	2 Mes 7 días - <b>#95</b>	20 Seg	ø

Figura 61 Trabajos para la ejecución de los pipelines.

Al finalizar la creación de las tareas se establece un orden de ejecución, para ello tenemos el plugin Build Pipeline.

### 3.4.1.2 Configuración del Plugin Build Pipeline

Jenkins posee una serie de complementos adicionales que permiten la manipulación de la ejecución de los trabajos, uno de ellos es "Build Pipeline" que permite visualizar la ejecución de los pipelines de una manera más gráfica.

Para utilizar este complemento se necesita instalarlo, dentro de las opciones de Panel de Control se selecciona Administrar Jenkins y se oprime un click sobre la etiqueta.



Figura 63 Opciones de System Configuration.

En el navegador se muestran un buscador y un conjunto de tabs que agrupan los plugin de acuerdo a las actualizaciones disponibles, todos los plugins, plugins instalados y una opcion de configuracion avanzada para configurar un plugin personalizado que no se encuentre en los repositorios de Jenkins, en el buscador se filtra por la opción Build Pipeline Plugin.

Q build						
Actualizacio	nes disponibles	Todos los plugins	Plugins instalados	Configuracion avanza	da	
Activados	Nombre 🛛				Versión	Versión previament
۲	Aqua MicroSca Enables sca	<b>nner</b> anning of docker build f	or OS package vulnerab	ilities.	1.0.8	
	Build Pipeline I This plugin	<b>Plugin</b> renders upstream and	downstream connected	jobs that typically		
	form a build for jobs that outside of .	d pipeline. In addition, i at require intervention Jenkins.	it offers the ability to de prior to execution, e.g. a	fine manual triggers n approval process		

Figura 64 Build Pipeline Plugin.

Para crear una vista Build Pipeline plugin se ingresa en la pantalla de inicio de Jenkins, en el centro del navegador se despliega en una grilla con todas las tareas creadas hasta el momento, en la cabecera se sitúa un tag con la etiqueta Todo y a un costado un tag con el signo "+" se oprime un click sobre este último.

	$\square$			
TesisMaestria	Todo +			
s	w 🔾	Nombre 1	Último Éxito	Último Fallo
	43	Curso DevOps	N/D	N/D
•	43	Docker	4 Mes 20 dias - #30	4 Mes 20 dias - <b>#29</b>

Figura 65 Opción para agregar pipelines.

La opción despliega un cuadro de texto para definir el nombre del pipeline, para este caso TesisMaestria y dos tipo de opciones Lista de visitas y Build Pipeline View se escoge este último.

🧌 Jenkins	Q búsqueda
Panel de Control 🕠	
쯜 Nueva Tarea	Nombre de vista
🍓 Personas	TesisMaestria
📄 Historial de trabajos	Shows the jobs in a build pipeline view. The complete pipeline of jobs that a version propagates through are shown as a row in the view.
🔍 Relacion entre proyectos	Lista de vistas Mostrar projectos como una lista simple. Puedes elegir qué proyectos mostrar en cada vista.
🚛 Comprobar firma de archivos	
참 Administrar Jenkins	ок
N estuali	

Figura 66 Creación de Pipeline.

El plugin funciona para enlazar trabajos en el orden que se desea para ello despliega en la pantalla una serie de opciones de configuración, en el centro debajo de la etiqueta Build Pipeline se encuentran un conjunto de acciones.

Run: Para ejecutar el pipeline e iniciar la ejecución de las tareas.

History: Muestra un registro de las ejecuciones del pipeline.

Configure: Muestra las configuraciones del pipeline, en esta sección se define la tarea inicial del pipeline.

Delete: Opción para borrar el pipeline.

Manage: Redirige al administrador de Jenkins.

🧌 Jenkins							<b>Q</b> búsqueda
Panel de Control 🔗 TesisMaestria 🔅							
	2 Run	Bu History	ild P	ipeli e Add Step	ine S Delete	2 Manage	

Figura 67 Opciones pipeline TesisMaestria.

Para configurar el primer trabajo a ejecutar se selecciona la opción Configure y se desplaza hasta la sección Pipeline Flow/Select initial jobs donde se ingresa el nombre de la tarea que será el primer trabajo a ejecutar en el pipeline, en este caso "CI\_Construccion".

# Upstream / downstream config

### Select Initial Job

### CI\_Construccion

Figura 68 Configuración del primer trabajo a ejecutar en la CI.

En este caso la CI\_Construccion inicia el flujo de toda la CI, para la configuración del orden de ejecución de las demás tareas se ingresa a un trabajo específico y se ingresa en su opción configurar, se marca la opción construir tras otros proyectos y se asigna el nombre del trabajo anterior a su ejecución.

Di	sparadores de ejecuciones	
<b>V</b>	Construir tras otros proyectos Projects to watch	?
	CI_PruebasSeguridad	
	Trigger only if build is stable	
	○ Trigger even if the build is unstable	
	○ Trigger even if the build fails	
	Consultar repositorio (SCM)	?
	Ejecutar periódicamente	?
	GitHub hook trigger for GITScm polling	?

Figura 69 Opción de Disparadores de Ejecución dentro de la tarea CI\_PruebasActualizacion.



Figura 70 Flujo de las tareas en la CI.

Jenkins posee una configuración llamada Disparadores de ejecuciones que sirve para programar el inicio de la ejecución de una tarea, en este caso se configura que cada 15 minutos se dispare la tarea CI\_Construccion.

Disparadores de ejecuciones	
Construir tras otros proyectos	0
Consultar repositorio (SCM)	?
Ejecutar periódicamente	()
Programador	0
H/15 * * * *	

Figura 71 Configuración de Disparadores de Ejecuciones.

### 3.4.1.3 Configuración de la tarea CI\_PruebasActualizacion

La tarea CI\_PruebasActualizacion se encarga de ejecutar las pruebas funcionales del servicio Empleado y las pruebas de seguridad sobre los contenedores que despliegan la aplicación, en esta configuración se asume que todas las herramientas se encuentran en el mismo servidor.

Como primer paso se debe recrear el funcionamiento de la Figura 40 dentro de Jenkins, como se mencionó en el capítulo 1 Diseño y arquitectura de la seguridad en un ambiente CI/CD Jenkins puede trabajar con repositorios, es decir se puede subir el archivo de docker compose dentro de un repositorio como Github y luego utilizarlo en Jenkins para construir los contenedores Docker con el servicio Empleado.

En este caso se necesita crear un repositorio GitHub con la estructura de directorio expuesta en la Figura 40.

🖿 app1	Actualizacion V.1
app2	Actualizacion V.1
B README.md	first commit
docker-compose.yaml	Actualizacion V1

Figura 72 Estructura de directorios en repositorio GitHub.

Una vez configurado y creado el repositorio en GitHub Jenkins debe detectarlo y utilizarlo dentro de una tarea, para que Jenkins descargue los archivos de Github a su plataforma y los utilice debemos seguir los siguientes pasos:

Se ingresa en la configuración de la tarea en la sección General y se

selecciona el tipo de proyecto, en este caso es GitHub project esta opción permite ingresar la url del proyecto Github https://github.com/DanielHur/Servicios.

General	Configurar el origen del código fuente	Disparadores de ejecuciones	Entorno de ejecución	Ejecutar
Acciones p	para ejecutar después.			
Descripción	1			
[Plain text]	Visualizar			ž
	t agent's Docker container			0
Define	a Docker template			
Desech	ar ejecuciones antiguas			?
🗆 Esta eje	ecución debe parametrizarse			?
🗹 GitHub	project			
Project	url			0
https://	/github.com/DanielHur/Servicios/			

Figura 73 Configuración de un proyecto GitHub.

El siguiente paso es indicar a Jenkins el origen del código fuente, en esta sección Jenkins tiene una opción de versionar el repositorio con la herramienta Git, debido a que la conexión entre Jenkins y GitHub es a través de HTTPS se debe especificar las credenciales para autenticarse al repositorio las mismas son creadas en la sección Credentials de Jenkins para luego utilizarlo en la tarea.

```
Configurar el origen del código fuente
```

epositories	
Repository URL	
https://github.com/DanielHur/Servicios.git	
Credentials	
hursonn@gmail.com/***** (Contrasena GitHub) ∨ 💽 🛃 Add マ	
	Avanzado.

Figura 74 Configuración del origen de código en Jenkins.



Figura 75 URL de conexión HTTPS Github.

La tarea CI\_PruebasActualizacion se conecta al repositorio Github y verifica si hubo un cambio en los directorios, en caso de ser así lo actualiza en el workspace asignado a la tarea en Jenkins, para este caso particular los archivos se bajan en la ruta:

/var/lib/jenkins/workspace/CI\_PruebasActualizacion.

Al tener los archivos del repositorio ya se tiene lo necesario para construir los contenedores con los servicios de Empleados a partir del docker compose, para ello nos desplazamos más abajo dentro de la configuración de la tarea hasta la sección de Ejecutar y se añade la sección "Ejecutar líneas de comando (shell)", esto despliega un cuadro donde podemos pasar comandos de consola desde la tarea de Jenkins.

E	jecutar
(	Añadir un nuevo paso 🔺
	Conditional step (single)
	Add a new template to all docker clouds Build / Publish Docker Image
l	Conditional steps (multiple)
(	Ejecutar linea de comandos (shell)
	Ejecutar tareas 'maven' de nivel superior
	Execute SonarQube Scanner
	Execute ZAP
	Invoke Gradle script

### Figura 76 Opción Ejecutar línea de comandos (shell).

Ej	ecutar	
	Ejecutar linea de comandos (shell)	× ?
	Comando	
	Visualizar la lista de variables de entorno disponibles	•
		Avanzado

Figura 77 Cuadro para insertar líneas de comandos.

Se ingresan las sentencias necesarias para instalar los contenedores y correr el servicio Empleado, para ello ejecutamos los siguientes comandos.

Primero se posiciona sobre la carpeta del workspace de la tarea.

cd /var/lib/jenkins/workspace/CI\_PruebasActualizacion

Y se ejecuta el comando docker-compose para construir.

docker-compose up -d --build

Esto inicia la construcción de los contenedores con el servicio Empleado, en el log de ejecución de la tarea en Jenkins se observa que en el espacio de trabajo /var/lib/jenkins/workspace/CI\_PruebasActualizacion se crean los contenedores de acuerdo al Dockerfile descargado desde Github.



Figura 78 Salida de la consola de Jenkins cuando construye los contenedores.

Al ejecutar el comando docker ps se observa 2 contenedores activos ci\_pruebasactualizacion\_app1 y ci\_pruebasactualizacion\_app2.

Figura 79 Contenedores en ejecución.

Los contenedores en este caso se despliegan con el nombre de la tarea de jenkins en minúscula en este caso ci\_pruebasactualizacion seguido de un guión bajo y el nombre del servicio definido en el docker compose app1 y app2.

Una vez levantados los contenedores iniciamos las pruebas funcionales para determinar si el servicio Empleado funciona como se espera en ambas versiones del contenedor. Como se mencionó anteriormente vamos a utilizar una herramienta llamada lightning para integrar las pruebas realizadas con Apache JMeter dentro de una CI.

### 3.4.1.3.1 Lightning

Lightning permite integrar las pruebas de Apache JMeter con el servidor CI/CD proporcionando información inequívoca para tomar decisiones autónomas si aprobar o rechazar el pipeline sin la necesidad de participación humana, si se considera el reporte generado por Apache Jmeter con los resultados de las pruebas de estrés sobre los servicios es posible generar un escenario donde estas pruebas determinan o no la continuidad del flujo del pipeline.

Si bien no es lo ideal la automatización de las pruebas de rendimiento debido a que se considera el factor humano un aspecto importante en su análisis, las pruebas realizadas en este caso sobre el servicio Empleado en ambos contenedores están basados en respuestas fijas de acuerdo a los parámetros que se le pasan; por lo que la funcionalidad del mismo se puede determinar por las respuestas satisfactorias que responden.

Lightning puede utilizarse tanto como un complemento dentro de Maven o descargarlo como un jar y utilizarlo de forma independiente, para este trabajo descargamos el JAR [14].

Clonamos el JAR desde su repositorio oficial en Git hub.

git clone https://github.com/automatictester/lightning-standalone-example.git

Figura 80 Comando para clonar Lightning desde el repositorio Git hub.

Lightning se utiliza para medir un conjunto de parámetros configurables por medio de un archivo xml en donde se definen los tipos de pruebas que pueden llevarse a cabo, en este trabajo analizamos las pruebas de transacciones aprobadas, para ello se configura un archivo xml llamado example.xml con la siguiente configuración [13].



Figura 81 Configuración xml para análisis con Lightning.

El tag allowedNumberOfFailedTransaction indica a la herramienta la cantidad de fallas que puede tolerar la aplicación antes de fallar en la integración, en este caso será cero ya que se busca que los servicios sean ejecutados de forma satisfactoria en ambos contenedores.

### 3.4.1.3.2 Lightning y Apache Jmeter

Para integrar las pruebas de Apache Jmeter con la CI el mismo debe ser ejecutado desde la línea de comandos, en este caso la herramienta puede ejecutarse desde la terminal a partir de un proyecto guardado como el de la figura 31 Apache Jmeter posee una forma de ejecutarse en modo no gráfico a través de comandos que tiene la siguiente forma:

jmeter.sh -n -t test-file [-p property-file] [-l results-file] [-j log-file]

Como Lightning trabaja con archivos de extensión csv es necesario que Apache JMeter construya la salida de su análisis en este formato, para lograr esto se debe crear un archivo de configuración adicional llamado jmeter.properties y se debe definir que la salida de la prueba sea un archivo de formato csv.

```
jmeter.save.saveservice.output_format=csv
jmeter.save.saveservice.print_field_names=true
jmeter.save.saveservice.successful=true
jmeter.save.saveservice.label=true
jmeter.save.saveservice.time=true
```

Figura 82 Configuración del archivo jmeter.properties.

Para ejecutar el Apache Jmeter y generar el archivo csv ejecutamos el siguiente comando.

sh jmeter.sh -n -t /var/lib/jenkins/workspace/CI\_PruebasActualizacion/JmeterPruebas/Emple ado.jmx -q /var/lib/jenkins/workspace/CI\_PruebasActualizacion/JmeterPruebas/jmeter. properties -I /home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv

El Apache Jmeter se ejecuta en modo no gráfico a través del comando " -n -t " luego se indica la ruta del proyecto de pruebas como en este caso el proyecto fue descargado del repositorio Git Hub este se encuentra alojado dentro del workspace generado por Jenkins con el mismo nombre de la tarea CI\_PruebasActualizacion "/var/lib/jenkins/workspace

/CI PruebasActualizacion/JmeterPruebas/Empleado.jmx", luego se indica la ubicación del archivo de configuración jmeter.properties "/var/lib/jenkins/ workspace/CI PruebasActualizacion/JmeterPruebas/jmeter.properties" y por indica dónde último se se va а generar el archivo csv /home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv con este comando ejecutamos el Apache Jmeter de acuerdo a la prueba generada por QA y generamos el archivo csv.

El siguiente paso es ejecutar Lightning a partir del csv generado por Apache Jmeter utilizando el archivo de configuración example.xml de la figura 81 y determinar el avance o la detención de la tarea CI\_PruebasActualizacion, para ello ejecutamos el siguiente comando.

echo Analiza reporte CSV

cd /home/dani/DockerServicio/lightning/lightning-standalone-example

java -jar bin/lightning-standalone-\*.jar verify

--xml=src/test/resources/xml/example.xml --jmeter-csv=/home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv

Para ejecutar Lightning ingresamos en la dirección donde se encuentra el JAR descargado del repositorio Figura 79 con el comando "cd /home/dani/DockerServicio/lightning/lightning-standalone-example", una vez ubicado en el directorio donde se encuentra Lightning ejecutamos el JAR y utilizamos el archivo example.xml de la Figura 80 con el comando "java -jar bin/lightning-standalone-\*.jar verify --xml=src/test/resources/

xml/example.xml" y se toma el archivo csv generado

"--jmeter-csv=/home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv" este comando analiza el archivo csv y si existe algún error detiene la CI.

Para visualizar cómo se comporta Lightning ante un error encontrado en el archivo CSV emulamos un error dentro de las pruebas.



Figura 83 Archivo PruebaJmLig.csv.

En este caso se produce un error al hacer un get a la aplicación empleado nos da un código HTTP 400, ejecutamos Lightning dentro de la tarea CI\_PruebasActualizacion en Jenkins para ello copiamos el comando expuesto anteriormente en la sección de Ejecutar Shell de Jenkins de la figura 76.



Figura 84 Comandos Lightning en Jenkins.

Si ejecutamos la tarea el log de Jenkins nos arroja el siguiente resultado.

+ echo Analiza repo	rte CSV
Analiza reporte CSV	
+ cd /home/dani/Doc	<pre>kerServicio/lightning/lightning-standalone-example</pre>
+ java -jar bin/lig	htning-standalone-7.0.0.jar verifyxml=src/test/resources/xml/example.xmljmeter-
csv=/home/dani/Dock	erServicio/ReportesTest/PruebaJmLig.csv
Test name:	Failed transactions
Test type:	passedTransactionsTest
Test description:	Verify number of passed tests
Expected result:	Number of failed transactions <= $0$
Actual result:	Number of failed transactions = 3872
Transaction count:	3872
Test result:	FAIL
====== EXECU.	TION SUMMARY ==========
Tests executed:	1
Tests passed:	9
Tests failed:	1
Tests errors:	9
Test set status:	FAIL
Execution time:	59ms
##teamcity[buildStat	tisticValue key='Failed transactions' value='3872']
Build step 'Ejecuta	r linea de comandos (shell)' marked build as failure
Finished: FAILURE	

Figura 85 Log de Jenkins al ejecutar Lightning con errores encontrados.

En la Figura 85 observamos que existe errores dentro de las pruebas de estrés para el servicio Empleado por lo que Lightning detiene el flujo de la CI. Si corregimos el test y no existen errores encontrados se muestra el siguiente resultado:

+ echo Analiza rep	orte CSV
Analiza reporte CS	1
+ cd /home/dani/Do	ckerServicio/lightning/lightning-standalone-example
+ java -jar bin/li	ghtning-standalone-7.0.0.jar verifyxml=src/test/resources/xml/example.xmljmeter-
csv=/home/dani/Doc	kerServicio/ReportesTest/PruebaJmLig.csv
Test name:	Failed transactions
Test type:	passedTransactionsTest
Test description:	Verify number of passed tests
Expected result:	Number of failed transactions <= $0$
Actual result:	Number of failed transactions = $0$
Transaction count:	30
Test result:	Pass
===== EXEC	JTION SUMMARY
Tests executed:	1
Tests passed:	1
Tests failed:	Θ
Tests errors:	Θ
Test set status:	Pass
Execution time:	28ms
##teamcity[buildSt	atisticValue key='Failed transactions' value='0']
Lanzando una nueva	ejecución de CI_PruebasDespliegue
Finished: SUCCESS	

Figura 86 Log de Jenkins al ejecutar Lightning con resultado exitoso.

En la Figura 86 se puede observar que el análisis fue exitoso y que a continuación se ejecutará la siguiente tarea CI\_PruebasDespliegue, de esta manera Lightning permite integrar las pruebas de Apache JMeter de acuerdo a las configuraciones que establecemos en el archivo example.xml expuesto en la Figura 81 continuar o detener la CI.

### 3.4.1.3.1 Trivy y Jenkins

Como se mencionó en el capítulo 3.3.2 Trivy el equipo de seguridad definió que solo las vulnerabilidades críticas dentro del contenedor en la versión latest detienen la tarea CI\_PruebasActualizacion, en esta etapa se va a generar un reporte de las vulnerabilidades encontradas y de acuerdo a la severidad se detendrá o se continuará con el pipeline.

A continuación se describen los pasos necesarios para ejecutar Trivy dentro de una CI:

Como primer paso se borra toda la información que Trivy podría tener en su caché sobre la imagen del contenedor de forma a volver a ejecutar un escaneo sobre la imagen con una base de datos de vulnerabilidades actualizadas y generar un reporte con los resultados arrojados, en caso de una vulnerabilidad crítica se detiene el pipeline, para ello ejecutamos el siguiente comando:

echo \*\*\*\*\*\*\*Borrando cache \*\*\*\*\*\*\* trivy --clear-cache ci\_pruebasactualizacion\_app1 echo \*\*\*\*\*\*\*Analiza vulnerabilidades \*\*\*\*\*\*\* trivy image --light --exit-code 1 --severity CRITICAL ci\_pruebasactualizacion\_app1 echo \*\*\*\*\*\*\*Exportando vulnerabilidades en Reporte \*\*\*\*\*\*\* trivy --output ReporteApp1 ci\_pruebasactualizacion\_app1

Al ejecutar estos comandos en Jenkins se produce la siguiente salida

### en el log.

********Borrando cache******* trivyclear-cache c1_pruebasactualizacion_app1 2021-08-29T16:51:55.76-0400 [[34mINF0][0m Removing image caches * echo ******Exportando vulnerabilidades en ReporteApp1 * trivyoutput ReporteApp1 c1_pruebasactualizacion_app1 2021-08-2716:51:55.71-04000 [[34mINF0][0m Downloading DB 10.60% 7 p/s 7720.84 K1B / 23.09 M1B [	+ echo *******Borrando cache******
<pre>+ triyclear-cache c1_pruebasctualizacion_app1 2201-08-2716:51:55:7.70-0400 [[34mINFO][0m Need to update D8 2201-08-2716:51:55:7.10-0400 [[34mINFO][0m Need to update D8 221-08-2716:51:55:7.10-0400 [[34mINFO][0m Need to update D8 22.03 MIB [ &gt; ] 5.03% 2.01 MIB p/S ETA 1052.03 MIB [23.09 MIB [ &gt; ] 5.03% 2.01 MIB p/S ETA 1052.03 MIB [23.09 MIB [ &gt; ] 10:95% 2.02 MIB p/S ETA 954.01 MIB / 23.09 MIB [ &gt; ] 13:09% 2.03 MIB p/S ETA 953.01 MIB / 23:09 MIB [&gt; ] 26:55% 2.02 MIB p/S ETA 954.01 MIB / 23:09 MIB [&gt; ] 26:55% 2.02 MIB p/S ETA 954.01 MIB p/S ETA 955.35 MIB / 23:09 MIB [&gt; ] 26:55% 2.02 MIB p/S ETA 954.01 MIB p/S ETA 955.35 MIB / 23:09 MIB [&gt; ] 26:55% 2.02 MIB p/S ETA 954.01 MIB p/S ETA 956.33 MIB p/S ETA 953.91 MIB / 23:09 MIB [&gt; ] 26:55% 2.02 MIB p/S ETA 954.01 MIB p/S ETA 956.33 MIB p/S ETA 856.99 MIB [&gt; ] 26:55% 2.01 MIB p/S ETA 950.07 MIB p/S ETA 856.99 MIB [&gt; ] 26:55% 2.01 MIB p/S ETA 756.02 MIB p/S ETA 956.39 MIB p/S ETA 856.99 MIB [] ] 26:25% 2.03 MIB p/S ETA 551.262 MIB p/S ETA 4514.29 MIB p/S ETA 758.08 MIB p/S ETA 4513.30 MIB p/S ETA 4513.30 MIB p/S ETA 4513.30 MIB p/S ETA 4513.30 MIB p/S ETA 4513.3</pre>	*******Borrando cache******
2021-08-29T16:51:55:667-0400 [[34mIRF0][0m Removing image caches + echo *******Exportando vulnerabilidades en ReporteAppl *******Exportando vulnerabilidades en ReporteAppl *******Exportando vulnerabilidades en ReporteAppl *******Exportando vulnerabilidades en ReporteAppl ********Exportando vulnerabilidades en ReporteAppl ********Exportando vulnerabilidades en ReporteAppl ***********************************	+ trivyclear-cache ci_pruebasactualizacion_app1
+ echo ********Exportando vulnerabilidades en ReporteAppl *******Exportando vulnerabilidades en ReporteAppl * trivyoutput ReporteAppl ci_pruebasactualizacion_appl 2021-08-29116:51:55.710-0400 []34mINF0][0m Need to update DB 2021-08-29116:51:55.710-0400 []34mINF0][0m Need to update DB 2021-08-29116:21:55.75.710-0400 []34mINF0][0m Need to update DB 2021-08-29116:21:51:55.710-0400 []34mINF0][0m Need to update DB 2021-08-29116:21:51:55.710-0400 []34mINF0][0m Need to update DB 2021-08-29116:21:51:55.710-0400 []34mINF0][0m Need to update DB 2021-08-203 MIB p/s ETA 1052.01 MIB / 23.09 MIB [ 2021-11:09% 2.03 MIB p/s ETA 953.02 MIB / 23.09 MIB [ 21:10:9% 2.02 MIB p/s ETA 953.01 MIB / 23.09 MIB [ 21:10:9% 2.02 MIB p/s ETA 953.01 MIB / 23.09 MIB [ 21:25.82% 2.11 MIB p/s ETA 953.01 MIB / 23.09 MIB [	2021-08-29T16:51:55.667-0400 [[34mINFO][0m Removing image caches
<pre>*******Exportando vulnerabilidades en ReporteApp1 * trivyoutput ReporteApp1 ci_pruebasctualizacion_app1 2021-08-29716:51:55.710-0400 []34mINF0][0m</pre>	+ echo *******Exportando vulnerabilidades en ReporteApp1
+ trivyoutput ReporteApp1 c1 pruebasactualizacion app1 2021-08-29T16:51:55.710-0400 [[34mINFO][0m Downloading DB 142.77 KIB / 23.09 MIB [>	*******Exportando vulnerabilidades en ReporteAppl
2021-08-29T16:51:55.710-0400       [[]34mINFO][0m       Need to update DB         2021-08-29T16:51:55.710-0400       [][34mINFO][0m       Downlog DB       ] 0.60% ? p/s ?729.84 K1B / 23.09 M1B [         >	+ trivyoutput ReporteApp1 ci_pruebasactualizacion_app1
2021-08-29T16:51:55.710-0400 [[34mINFO[]0m Downloading DB 142.77 KIB / 23.09 MIB [>] 0.60%; ? p/s 71.35 MIB / 23.09 MIB [ >] 5.83%; ? p/s 71.35 MIB / 23.09 MIB [ >] 5.83%; ? p/s 71.35 MIB / 23.09 MIB [ >] 5.83%; ? p/s 71.35 MIB / 23.09 MIB [ >] 5.83%; ? p/s 71.35 MIB / 23.09 MIB [ >] 5.83%; ? p/s 71.35 MIB / 23.09 MIB [ >] 5.83%; ? p/s 71.35 MIB / 23.09 MIB [ >] 5.83%; ? p/s 71.35 MIB / 23.09 MIB [ >] 9.95%; 2.01 MIB p/s ETA 1052.36 MIB / 23.09 MIB [ >] 10.90%; ? 0.3 MIB p/s ETA 1053.02 MIB / 23.09 MIB [ >] 10.90%; 2.03 MIB p/s ETA 953.44 MIB / 23.09 MIB [ >] 11.90%; 2.03 MIB p/s ETA 953.44 MIB / 23.09 MIB [ >] 10.95%; 2.02 MIB p/s ETA 954.61 MIB / 23.09 MIB [ >] 25.62%; 2.11 MIB p/s ETA 954.61 MIB / 23.09 MIB [ >] 25.62%; 2.11 MIB p/s ETA 856.03 MIB / 23.09 MIB [ >] 25.62%; 2.11 MIB p/s ETA 856.03 MIB / 23.09 MIB [ >] 25.62%; 2.11 MIB p/s ETA 856.03 MIB / 23.09 MIB [ >] 25.62%; 2.03 MIB p/s ETA 856.09 MIB / 23.09 MIB [ >] 26.52%; 2.03 MIB p/s ETA 756.09 MIB [ >] 26.52%; 2.03 MIB p/s ETA 756.09 MIB [ >] 26.62%; 2.03 MIB p/s ETA 756.09 MIB [ >] 30.00%; 2.03 MIB p/s ETA 756.09 MIB [ >] 30.20%; 2.03 MIB p/s ETA 756.09 MIB [ >] 30.90%; 2.03 MIB p/s ETA 756.09 MIB [ >] 30.20%; 2.03 MIB p/s ETA 756.09 MIB [ >] 30.90%; 2.03 MIB p/s ETA 756.09 MIB [ >] 30.90%; 2.03 MIB p/s ETA 551.06 MIB / 23.09 MIB [ >] 30.90%; 2.03 MIB p/s ETA 551.06 MIB / 23.09 MIB [ >] 27.30	2021-08-29T16:51:55.710-0400 [[34mINFO][0m Need to update DB
142.77 KiB / 23.09 MiB [>] 0.60% 7 p/s 7729.84 KiB / 23.09 MiB []         3.09% 7 p/s 71.35 MiB / 23.09 MiB []         5.83% 7 p/s 71.35 MiB / 23.09 MiB []         5.83% 7 p/s 71.35 MiB / 23.09 MiB []         5.83% 7 p/s 71.35 MiB / 23.09 MiB []         9.05% 2.01 MiB p/s ETA 1052.20 MiB / 23.09 MiB []         9.05% 2.01 MiB p/s ETA 1053.02 MiB / 23.09 MiB []         9.05% 2.01 MiB p/s ETA 1053.02 MiB / 23.09 MiB [         11.90% 2.03 MiB p/s ETA 933.44 MiB / 23.09 MiB [         13.09% 2.03 MiB p/s ETA 933.91 MiB / 23.09 MiB [>         14.90% 2.03 MiB p/s ETA 933.91 MiB / 23.09 MiB [>         12.5% 2.02 MiB p/s ETA 954.61 MiB / 23.09 MiB [>         12.5% 2.02 MiB p/s ETA 856.33 MiB / 23.09 MiB [>         12.5% 2.11 MiB p/s ETA 856.33 MiB / 23.09 MiB [>         12.5% 2.11 MiB p/s ETA 856.33 MiB / 23.09 MiB [>         12.6.2% 2.11 MiB p/s ETA 856.33 MiB / 23.09 MiB [>         12.6.2% 2.11 MiB p/s ETA 856.30 MiB p/s ETA 656.90 MiB / 23.09 MiB [>         12.6.2% 2.11 MiB p/s ETA 757.62 MiB / 23.09 MiB [>         13.6.2% 2.03 MiB p/s ETA 659.59 MiB / 23.09 MiB [>         13.6.2% 2.03 MiB p/s ETA 659.59 MiB / 23.09 MiB [>         13.6.2% 2.03 MiB p/s ETA 659.59 MiB / 23.09 MiB [>         13.6.2% 2.03 MiB p/s ETA 659.59 MiB / 23.09 MiB [>         13.6.2% 2.03 MiB p/s	2021-08-29T16:51:55.710-0400 [[34mINFO][0m Downloading DB
>         ] 3.09% ? 0 / 5 7 / 35 MiB / 23.09 MiB [           >         ] 5.03% 2.01 MiB p/s ETA 1052.01 MiB / 23.09 MiB [           >         ] 8.71% 2.01 MiB p/s ETA 1052.03 MiB / 23.09 MiB [           >         ] 9.95% 2.01 MiB p/s ETA 1053.02 MiB / 23.09 MiB [           >         ] 11.90% 2.03 MiB p/s ETA 1053.02 MiB / 23.09 MiB [           >         ] 11.90% 2.03 MiB p/s ETA 953.44 MiB / 23.09 MiB [           >         ] 13.09% ? 0.03 MiB p/s ETA 953.44 MiB / 23.09 MiB [           >         ] 14.90% 2.03 MiB p/s ETA 953.44 MiB / 23.09 MiB [           >         ] 25.82% 2.11 MiB p/s ETA 953.41 MiB / 23.09 MiB [	142.77 KiB / 23.09 MiB [>] 0.60% ? p/s ?729.84 KiB / 23.09 MiB [-
>	> ] 3.09% ? p/s ?1.35 MiB / 23.09 MiB [
>       ] 5.83% 2.01 MiB p/s ETA 10s2.01 MiB / 23.09 MiB [         >       ] 8.71% 2.01 MiB p/s ETA 10s2.25.30 MiB / 23.09 MiB [         >       ] 9.95% 2.01 MiB p/s ETA 10s2.75 MiB / 23.09 MiB [         >       ] 11.90% 2.03 MiB p/s ETA 10s3.02 MiB / 23.09 MiB [         >       ] 11.90% 2.03 MiB p/s ETA 10s3.02 MiB / 23.09 MiB [         >       ] 11.90% 2.03 MiB p/s ETA 9s3.301 MiB / 23.09 MiB [	> ] 5.83% ? p/s ?1.35 MiB / 23.09 MiB [
>	>] 5.83% 2.01 MiB p/s ETA 10s2.01 MiB / 23.09 MiB [
>       ] 9.95% 2.01 MiB p/s ETA 1053.02 MiB / 23.09 MiB [         >       ] 11.90% 2.03 MiB p/s ETA 953.04 MiB / 23.09 MiB [         >       ] 13.09% 2.03 MiB p/s ETA 953.04 MiB / 23.09 MiB [         >       ] 14.90% 2.03 MiB p/s ETA 953.01 MiB / 23.09 MiB [         >       ] 14.90% 2.03 MiB p/s ETA 953.91 MiB / 23.09 MiB [	>] 8.71% 2.01 MiB p/s ETA 10s2.30 MiB / 23.09 MiB [
>       11.99% 2.03 MIB p/s ETA 1953.02 MIB / 23.09 MIB [         >       13.09% 2.03 MIB p/s ETA 953.02 MIB / 23.09 MIB [         >       14.90% 2.03 MIB p/s ETA 953.01 MIB / 23.09 MIB [	>] 9.95% 2.01 MiB p/s ETA 10s2.75 MiB / 23.09 MiB [
>       ] 13.09% 2.03 MIB p/s ETA 953.91 MIB / 23.09 MIB [	>] 11.90% 2.03 MiB p/s ETA 10s3.02 MiB / 23.09 MiB [
>       1 44.90% 2.03 MiB p/s ETA 9s3.91 MiB / 23.09 MiB [	>] 13.09% 2.03 MiB p/s ETA 9s3.44 MiB / 23.09 MiB [
16.95% 2.02 MiB p/s ETA 9s4.61 MiB / 23.09 MiB [	>] 14.90% 2.03 MiB p/s ETA 9s3.91 MiB / 23.09 MiB [>]
MIB       [ 23.19% 2.02 MIB p/s ETA 855.96 MIB / 23.09 MIB       [	16.95% 2.02 MiB p/s ETA 9s4.61 MiB / 23.09 MiB [>] 19.95% 2.02 MiB p/s ETA 9s5.35 MiB / 23.09
>         ]         25.82%         2.11 MiB p/s ETA 856.12 MiB / 23.09 MiB []         ]         27.94%         2.01 MiB p/s ETA 866.33 MiB / 23.09 MiB []         ]         27.43%         2.11 MiB p/s ETA 756.44 MiB / 23.09           NIB	MiB [>] 23.19% 2.02 MiB p/s ETA 8s5.96 MiB / 23.09 MiB [
26.52% 2.11 MiB p/s ETA 8s6.33 MiB / 23.09 MiB [	>] 25.82% 2.11 MiB p/s ETA 856.12 MiB / 23.09 MiB [>]
MIB       [	26.52% 2.11 MiB p/s ETA 8s6.33 MiB / 23.09 MiB [>] 27.43% 2.11 MiB p/s ETA 7s6.44 MiB / 23.09
>         1         28.87%         2.03 MiB p/s ETA 856.99 MiB / 23.09 MiB         1         33.06%         2.03 MiB p/s ETA 757.62 MiB / 23.09 MiB [	MiB [>] 27.90% 2.03 MiB p/s ETA 8s6.67 MiB / 23.09 MiB [
2.03 MiB p/s ETA 757.62 MiB / 23.09 MiB [	>] 28.87% 2.03 MiB p/s ETA 8s6.99 MiB / 23.09 MiB [>] 30.28%
	2.03 MiB p/s ETA 7s7.62 MiB / 23.09 MiB [] 33.00% 2.03 MiB p/s ETA 7s8.38 MiB / 23.09 MiB [
>         ] 38.94% 2.03 MiB p/s ETA 659.59 MiB / 23.09 MiB [] 41.52% 2.11           MiB p/s ETA 6510.31 MiB / 23.09 MiB [] 47.83% 2.11 MiB p/s ETA 5511.83 MiB / 23.09 MiB [] 44.64% 2.11 MiB p/s ETA 6511.04 MiB / 23.09 MiB [] 51.24% 2.21 MiB p/s ETA 5512.62 MiB / 23.09 MiB [] 58.29% 2.21 MiB p/s ETA 5512.62 MiB / 23.09 MiB [] 54.67% 2.21 MiB p/s ETA 4s13.46 MiB / 23.09           MiB [] 61.88% 2.33 MiB p/s ETA 3515.08 MiB / 23.09 MiB [] 58.29% 2.21 MiB p/s ETA 4s14.29 MiB / 23.09 MiB [] 65.34% 2.33 MiB p/s ETA 3515.87 MiB / 23.09 MiB [] 65.75% 2.33 MiB p/s ETA 3516.71 MiB / 23.09 MiB [] 75.58% 2.44           MiB p/s ETA 2518.30 MiB / 23.09 MiB [] 72.38% 2.44 MiB p/s ETA 2517.45 MiB / 23.09 MiB [] 79.27% 2.44 MiB p/s ETA 1519.09 MiB [] 75.58% 2.44           MiB p/s ETA 1520.32 MiB / 23.09 MiB [] 88.47% 2.52 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 88.47% 2.52 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.52 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MiB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MIB [] 188.47% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MIB [] 188.47\% 2.54 MiB p/s ETA 1520.43 MiB / 23.09 MIB [] 188.47\% 2.54 MiB p/s ETA 1520.43 MiB / 23.	>] 36.29% 2.03 MiB p/s ETA 7s8.99 MiB / 23.09 MiB [
MIB p/s ETA 6s10.31 MIB / 23.09 MIB []       44.64% 2.11 MIB p/s ETA 6s11.04 MIB / 23.09 MIB []         S1.24% 2.21 MIB p/s ETA 5s12.62 MIB / 23.09 MIB []       54.67% 2.21 MIB p/s ETA 4s13.46 MIB / 23.09         MIB []       54.67% 2.21 MIB p/s ETA 4s13.46 MIB / 23.09         MIB []       54.67% 2.21 MIB p/s ETA 4s13.46 MIB / 23.09         MIB []       54.67% 2.21 MIB p/s ETA 4s13.46 MIB / 23.09         MIB []       58.29% 2.21 MIB p/s ETA 4s14.29 MIB / 23.09 MIB []         S15.87 MIB / 23.09 MIB []       68.75% 2.33 MIB p/s ETA 3s16.71 MIB / 23.09 MIB []         S15.87 MIB / 23.09 MIB []       72.38% 2.44 MIB p/s ETA 2s17.45 MIB / 23.09 MIB []         MIB p/s ETA 2s18.30 MIB / 23.09 MIB []       175.58% 2.44         MIB p/s ETA 1s10.91 MIB / 23.09 MIB []       182.70% 2.54 MIB p/s ETA 1s19.91 MIB / 23.09 MIB []         86.25% 2.54 MIB p/s ETA 1s20.32 MIB / 23.09 MIB []       188.47% 2.52 MIB p/s ETA 1s20.43 MIB / 23.09 MIB []	>] 38.94% 2.03 MiB p/s ETA 6s9.59 MiB / 23.09 MiB [>] 41.52% 2.11
	MiB p/s ETA 6s10.31 MiB / 23.09 MiB [] 44.64% 2.11 MiB p/s ETA 6s11.04 MiB / 23.09 MiB [
51.24% 2.21 MiB p/s ETA 5512.62 MiB / 23.09 MiB [] 54.67% 2.21 MiB p/s ETA 4s13.46 MiB / 23.09 MiB [] 58.29% 2.21 MiB p/s ETA 4s14.29 MiB / 23.09 MiB [] 65.34% 2.33 MiB p/s ETA 3s15.08 MiB / 23.09 MiB [] 65.34% 2.33 MiB p/s ETA 3s15.08 MiB / 23.09 MiB [] 65.75% 2.33 MiB p/s ETA 3s16.71 MiB / 23.09 MiB [] 72.38% 2.44 MiB p/s ETA 2s17.45 MiB / 23.09 MiB [] 72.38% 2.44 MiB p/s ETA 2s17.45 MiB / 23.09 MiB [] 75.58% 2.44         MIB p/s ETA 2s18.30 MIB / 23.09 MIB [] 72.38% 2.44 MiB p/s ETA 2s17.45 MiB / 23.09 MiB [] 75.58% 2.44         MIB p/s ETA 2s18.30 MIB / 23.09 MIB [] 72.58% 2.44 MiB p/s ETA 1s19.91 MiB / 23.09 MiB [] 82.70% 2.54 MIB p/s ETA 1s19.91 MIB / 23.09 MIB [] 88.00% 2.54 MiB p/s ETA 1s20.43 MiB / 23.09         MIB [] 88.47% 2.52 MiB p/s ETA 1s20.43 MiB / 23.09 MIB [] 88.47% 2.52 MIB p/s ETA 1s20.43 MIB / 23.09 MIB [] 75.00% MIB / 23.09	> ] 47.83% 2.11 MiB p/s ETA 5s11.83 MiB / 23.09 MiB [> ]
M1B       []       58.29%       2.21       M1B       p/s       ETA       4514.29       M1B       [	51.24% 2.21 MiB p/s ETA 5s12.62 MiB / 23.09 MiB [>] 54.67% 2.21 MiB p/s ETA 4s13.46 MiB / 23.09
	MiB [>] 58.29% 2.21 MiB p/s ETA 4s14.29 MiB / 23.09 MiB [
3s15.87 MiB / 23.09 MiB [	>] 61.88% 2.33 MiB p/s ETA 3s15.08 MiB / 23.09 MiB [>] 65.34% 2.33 MiB p/s ETA
	3s15.87 MiB / 23.09 MiB [>] 68.75% 2.33 MiB p/s ETA 3s16.71 MiB / 23.09 MiB [
MIB p/s ETA 2s18.30 MIB / 23.09 MIB [] 79.27% 2.44 MIB p/s ETA 1s19.09 MIB / 23.09 MIB [	>] 72.38% 2.44 MiB p/s ETA 2s17.45 MiB / 23.09 MiB [>] 75.58% 2.44
	MiB p/s ETA 2s18.30 MiB / 23.09 MiB [>] 79.27% 2.44 MiB p/s ETA 1s19.09 MiB / 23.09 MiB [
86.25% 2.54 MiB p/s ETA 1s20.32 MiB / 23.09 MiB [>] 88.00% 2.54 MiB p/s ETA 1s20.43 MiB / 23.09 MiB [>] 88.47% 2.52 MiB p/s ETA 1s20.43 MiB / 23.09 MiB [>]	>] 82.70% 2.54 MiB p/s ETA 1s19.91 MiB / 23.09 MiB [
M1B [> ] 88.47% 2.52 M1B p/s ETA 1s20.43 M1B / 23.09 M1B [	86.25% 2.54 MiB p/s ETA 1s20.32 MiB / 23.09 MiB [] 88.00% 2.54 MiB p/s ETA 1s20.43 MiB / 23.09
	MIB [> ] 88.47% 2.52 MIB p/s ETA 1s20.43 MIB / 23.09 MIB [

\_] 88.47% 2.52 MiB p/s ETA 1s20.43 MiB / 23.09 MiB [------> ] 88.47% 2.52 MiB p/s ETA 1s20.81 ] 90.14% 2.40 MiB p/s ETA 0s21.09 MiB / 23.09 MiB [-----MiB / 23.09 MiB [-----] 93.09% 2.40 MiB p/s >] 97.00% 2.37 MiB p/s ETA 0s22.03 MiB / 23.09 MiB [------>] 98.86% 2.37 MiB p/s ETA 0s23.09 MiB / 23.09 MiB [------>] 100.00% 2.57 MiB p/s 9s2021-08-29T16:53:01.167- 

 2.37 MiB p/S ETA 0523.09 MiB / 23.09 MiB / 23.0 ci\_pruebasactualizacion\_app1 (alpine 3.10.9) Total: 1 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 1) usr/src/servicio/rest-0.0.1-SNAPSHOT.jar Total: 10 (UNKNOWN: 0, LOW: 0, MEDIUM: 3, HIGH: 6, CRITICAL: 1) + echo \*\*\*\*\*\*\*Analiza vulnerabilidades\*\*\*\*\*\* \*\*\*\*\*Analiza vulnerabilidades\* + trivy image --light --exit-code 1 --severity CRITICAL ci pruebasactualizacion app1 2021-08-29T16:53:01.435-0400 [[34mINFO][0m Need to update DB 2021-08-29T16:53:01.436-0400 [[34mINFO][0m Downloading DB... \_] 54.47% 3.36 MiB p/s ETA 0s3.11 MiB / 5.72 MiB [-----\_] 54.47% 3.36 MiB p/s ETA 0s3.78 MiB / 5.72 MiB [------] 66.09% 3.32 MiB p/s ETA 0s5.47 MiB / 5.72 MiB [-----] 95.74% 3.32 MiB p/s ETA 055.72 MiB / 5.72 MiB [-----] 100.00% 3.72 MiB p/s 2s2021-08-29T16:53:10.429-0400 ci\_pruebasactualizacion\_app1 (alpine 3.10.9) Total: 1 (CRITICAL: 1) | LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION | | apk-tools | CVE-2021-36159 | CRITICAL | 2.10.6-r0 | 2.10.7-r0 | usr/src/servicio/rest-0.0.1-SNAPSHOT.jar Total: 1 (CRITICAL: 1) \*-----\* LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION | 1 +----+ | net.minidev:json-smart | CVE-2021-27568 | CRITICAL | 2.3 | 2.4.1, 1.3.2 | Build step 'Ejecutar linea de comandos (shell)' marked build as failure Finished: FAILURE

Figura 87 Log de Jenkins al ejecutar Trivy.

+   LIBRARY	+	++   SEVERITY	INSTALLED VERSION	FIXED VERSION	•	тітle	
apk-tools     	k-tools (CVE-2021-36159 CRITICAL 2.		2.12.5-r1		libfetch before 2021-07-26, as   used in apk-tools, xbps, and   other products, mishandles  >avd.aquasec.com/nvd/cve-2021-36159		
libcrypto1.1   	ltbcryptol.1 CVE-2021-3711 HIGH 1.1 CVE-2021-3712 MEDIUM Ltbssll.1 CVE-2021-3711 HIGH		1.1.1k-r0		openssl: SM2 Decryption     Buffer Overflow   savd.aguase.com/nvd/cve-2021-3711		
+ · ·     					openssl: Read buffer overruns   processing ASN.1 strings  >avd.aquasec.com/nvd/cve-2021-3712		
libssl1.1   					openssl: SM2 Dec   Buffer Overflow  >avd.aquasec.c	ryption     om/nvd/cve-2021-3711	.on
+ · · · · · · · · · · · · · · · · · · ·	CVE-2021-3712	MEDIUM   	-     		openssl: Read buffer overruns   processing ASN.1 strings >avd.aquasec.com/nvd/cve-2021-3712		
+ I	LIBRARY		VULNERABILITY I	D   SEVERITY	INSTALLED VERSION	+ FIXED VERSION	
net.minidev:json-smart     			CVE-2021-27568   	CRITICAL       		2.4.1, 1.3.2   	json-smart: uncaught     exception may lead to crash     or information disclosure    >avd.aquasec.com/nvd/cve-2021-27568
org.apache.toncat.enbed:toncat-enbed-core			CVE-2020-13934	HIGH	9.0.35	8.5.57, 9.0.37     	tomcat: OutOfMemoryException     caused by HTTP/2 connection     leak could lead to DoS    >avd.aquasec.com/nvd/cve-2020-13934
			CVE-2020-17527 			8.5.60, 9.0.40, 10.0.2 	tomcat: HTTP/2 request header mix-up    >avd.aquasec.com/nvd/cve-2020-17527
			CVE-2021-25122 			8.5.63, 9.0.43, 10.0.2 	tomcat: Request mix-up with h2c    >avd.aquasec.com/nvd/cve-2021-25122
			CVE-2021-25329			7.0.108, 8.5.61, 9.0.41,   10.0.2 	tomcat: Incomplete fix     for CVE-2020-9484 (RCE     via session persistence)    >avd.aquasec.com/nvd/cve-2021-25329
+     			CVE-2021-24122	MEDIUM		7.0.107, 8.5.60, 9.0.40,   10.0.0-M10 	tomcat: Information disclosure     when using NTFS file system    >avd.aquasec.com/nvd/cve-2021-24122
 +				+			>avd.aquasec.com/nvd/cve-2021-24122   +
org.apache.tom     	ncat.embed:tomcat-er	nbed-websock				7.0.105, 8.5.57, 9.0.37     	tomcat: multiple requests     with invalid payload length     in a WebSocket Frame could    >avd.aquasec.com/nvd/cve-2020-13935
+     			CVE-2021-24122   	MEDIUM   		10.0.0-M10, 9.0.40, 8.5.60,   7.0.107 	tomcat: Information disclosure   when using NTFS file system  >avd.aquasec.com/nvd/cve-2021-24122
org.glassfish:jakarta.el   			CVE-2021-28170		3.0.3		jakarta-el: ELParserTokenManager     enables invalid EL     expressions to be evaluate    >avd.aquasec.com/nvd/cve-2021-28170
org.hibernate:hibernate-core     			CVE-2020-25638	HIGH	5.4.15.Final	5.3.20.Final, 5.4.24.Final     	hibernate-core: SQL injection     vulnerability when both     hibernate.use_sqL_comments and     JPQL String literals are  savd.aquasec.com/nvd/cve-2020-25638

Figura 88 Reporte de vulnerabilidades Trivy.

En este caso se detiene la tarea debido a que el contenedor en la versión latest posee una vulnerabilidad crítica, por lo que las pruebas de actualización serán detenidas.

### 4. Implementación de las pruebas de actualización

Una vez desarrollado todos los pasos necesarios para las pruebas de actualización es momento de unir todos los procesos en una sola tarea de Jenkins, para ello nos disponemos a escribir los comandos necesarios dentro de la herramienta en la sección de Ejecutar/Ejecutar líneas de comandos (shell) Figura 76:

Resumimos los pasos que debe realizar la CI\_PruebasActualizacion:

1. Levantamos los contenedores con el servicio Empleado en ambas versiones, uno en latest y otro en la versión de

producción alpine:3.10.

- 2. Aplicamos las pruebas elaboradas en Jmeter para generar los informes y generar el archivo csv.
- 3. Ejecutamos Lightning para analizar los resultados arrojados de las pruebas con Apache Jmeter.
- 4. Realizamos el test de vulnerabilidades sobre el contenedor en la versión latest.



Figura 89 Comandos de ejecución para la tarea CI\_PruebasActualizacion.



echo ******Generar Archivo csv con Apache JMeter aplicacion version latest****** sh jmeter.sh -n -t /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Emple ado.jmx -q /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/jmeter. properties -I /home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv
echo ******Generar Archivo csv con Apache JMeter aplicacion version produccion***** sh jmeter.sh -n -t /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Emple ado2.jmx -q /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/jmeter. properties -I /home/dani/DockerServicio/ReportesTest/PruebaJmLig1.csv
echo ******Analisis de aplicacion contenedor Version Produccion****** cd /home/dani/DockerServicio/lightning/lightning-standalone-example java -jar bin/lightning-standalone-*.jar verify xml=src/test/resources/xml/example.xml jmeter-csv=/home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv
echo ******Analisis de aplicacion contenedor Version latest***** java -jar bin/lightning-standalone-*.jar verify xml=src/test/resources/xml/example.xml jmeter-csv=/home/dani/DockerServicio/ReportesTest/PruebaJmLig1.csv
echo *******ANALISIS DE VUNERABILIDADES****** echo *******Borrando cache******* trivyclear-cache ci_pruebasactualizacion_app1
echo *******Exportando vulnerabilidades en Reporte******* trivyoutput ReporteApp1 ci_pruebasactualizacion_app1
echo *******Analiza vulnerabilidades****** trivy imagelightexit-code 1severity CRITICAL ci_pruebasactualizacion_app1

Ejecutamos la tarea CI\_PruebasActualizacion y observamos los logs.



Figura 90. Opción de Jenkins para construir la tarea.

Salida del Log de Jenkins de la tarea CI\_PruebasActualizacion.

# Salida de consola

Lanzada por el usuario unknown or anonymous Running as SYSTEM Ejecutando.en el espacio de trabajo /var/lib/jenkins/workspace/CI\_PruebasActualizacion The recommended git tool is: NONE The recommended git tool 15: NUNE using credential f8a0be8a-e6f9-4977-babb-ef57e6e2d3df > git rev-parse --is-inside-work-tree # timeout=10 Fetching changes from the remote Git repository > git config remote.origin.url https://github.com/DanielHur/Servicios.git # timeout=10 Fetching upstream changes from https://github.com/DanielHur/Servicios.git
> git --version # timeout=10 > git --version # 'git version 2.25.1' using GIT\_ASKPASS to set credentials Contrasena GitHub > git feth -tags - force -rprogress - https://github.com/DanielHur/Servicios.git +refs/heads/\*:refs/remotes/origin/\* # timeout=10 > git rev-parse refs/remotes/origin/main^{commit} # timeout=10 Checking out Revision 4a0cd76b6e2cd2aa9c562cbf6e8ea122d24acf8b (refs/remotes/origin/main) > git config core.sparsecheckout # timeout=10 > git conig core.spase.reckout # timeout=10
> git checkout -f 4a0cd76b6e2cd2aa9c562cbf6e8ea122d24acf8b # timeout=10
Commit message: "Actualizacion V1"
> git rev-list --no-walk 4a0cd76b6e2cd2aa9c562cbf6e8ea122d24acf8b # timeout=10 [CI\_PruebasActualizacion] \$ /bin/sh -xe /tmp/jenkins10674894234293231818.sh + cd /var/lib/jenkins/workspace/CI PruebasActualizacion + docker-compose up -d --build Building app1 Step 1/7 : FROM alpine:latest ---> d4ff818577bc Step 2/7 : RUN apk add openjdk11 ---> Using cache --> bd6fdb191b78 Step 3/7 : ENV JAVA\_HOME="/usr/lib/jvm/default-jvm/" ---> Using cache ---> 38127fd4cbcc
```
Step 4/7 : ENV PATH=$PATH:${JAVA HOME}/bin
 ---> Using cache
 ---> 42f7e4e66923
Step 5/7 : COPY . /usr/src/servicio
 ---> Using cache
---> adfb82ea5fb8
Step 6/7 : WORKDIR /usr/src/servicio
 ---> Using cache
 ---> 708c6dd71d93
Step 7/7 : CMD [ "java", "-jar", "rest-0.0.1-SNAPSHOT.jar" ]
---> Using cache
---> 579719b69aff
Successfully built 579719b69aff
Successfully tagged ci_pruebasactualizacion_app1:latest
Building app2
Step 1/7 : FROM alpine:3.10.9
 ---> e7b300aee9f9
Step 2/7 : RUN apk add openjdk11
 ---> Using cache
---> fb26fba87460
Step 3/7 : ENV JAVA_HOME="/usr/lib/jvm/default-jvm/"
 ---> Using cache
---> bceb12dda77d
Step 4/7 : ENV PATH=$PATH:${JAVA HOME}/bin
---> Using cache
 ---> 36f6ae10de14
Step 5/7 : COPY . /usr/src/servicio
---> Using cache
---> cbcd0d743a41
Step 6/7 : WORKDIR /usr/src/servicio
---> Using cache
 ---> bd2288970c44
Step 7/7 : CMD [ "java", "-jar", "rest-0.0.1-SNAPSHOT.jar" ]
 ---> Using cache
 ---> 4e53f1028b62
Successfully built 4e53f1028b62
Successfully tagged ci_pruebasactualizacion_app2:latest
```

```
ci_pruebasactualizacion_app2_1 is up-to-date
ci_pruebasactualizacion_app1_1 is up-to-date
+ sleep 5
         + echo
*******ANALISIS DE FUNCIONALIDAD*******
+ echo ******Generar Archivo csv con Apache JMeter******
******Generar Archivo csv con Apache JMeter******
+ cd /home/dani/apache-jmeter-5.3/bin
+ sh imeter.sh -- jmeter.save.saveservice.output format=xml -n -t /var/lib/jenkins/workspace/CI PruebasActualizacion/JmeterPruebas/Empleado.jmx
l /home/dani/Servicios/Servicios/JmeterPruebas/TestVersionLatest.jtl
Creating summariser <summarv>
Created the tree successfully using /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Empleado.jmx
Starting standalone test @ Tue Aug 31 17:25:20 PYT 2021 (1630445120920)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary = 30 in 00:00:02 = 15,9/s Avg: 24 Min: 3 Max: 128 Err: 0
                                                                                                   0 (0,00%)
Tidying up ...
... end of run
                    @ Tue Aug 31 17:25:23 PYT 2021 (1630445123381)
... end of num
+ sh jmeter.sh -Jjmeter.save.saveservice.output_format=xml -n -t /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Empleado2.jmx
-l /home/dani/Servicios/Servicios/JmeterPruebas/TestVersionProd.jtl
Creating summariser <summary>
Created the tree successfully using /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Empleado2.jmx
Starting standalone test @ Tue Aug 31 17:25:24 PYT 2021 (1630445124905)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
                30 in 00:00:02 = 15,6/s Avg: 33 Min: 3 Max
.. @ Tue Aug 31 17:25:27 PYT 2021 (1630445127184)
summary =
                                                                        3 Max: 252 Err:
                                                                                                    0 (0.00%)
Tidying up ...
+ sh jmeter.sh -n -t /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Empleado.jmx -q
/var/lib/jenkins/workspace/CI PruebasActualizacion/JmeterPruebas/jmeter.properties -l /home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv
Creating summariser <summary>
Created the tree successfully using /var/lib/jenkins/workspace/CI PruebasActualizacion/JmeterPruebas/Empleado.jmx
Starting standalone test @ Tue Aug 31 17:25:28 PYT 2021 (1630445128780)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary + 13 in 00:00:01 = 15,0/5 Avg: 11 Min: 3 Max: 46 Err: 0 (0,00%) Active: 1 Started: 10 Finished: 9
summary + 17 in 00:00:01 = 16,3/5 Avg: 16 Min: 9 Max: 25 Err: 0 (0,00%) Active: 0 Started: 20 Finished: 20
summary = 30 in 00:00:02 = 15,7/5 Avg: 14 Min: 3 Max: 46 Err: 0 (0,00%)
Tidying up ... @ Tue Aug 31 17:25:31 PYT 2021 (1630445131064)
... end of run
+ echo ******Generar Archivo csv con Apache JMeter aplicacion version produccion******
*******Generar Archivo csv con Apache JMeter aplication version produccion*******
+ sh jmeter.sh -n -t /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Empleado2.jmx -q
/var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/jmeter.properties -l /home/dani/DockerServicio/ReportesTest/PruebaJmLigl.csv
Creating summariser <summary>
Created the tree successfully using /var/lib/jenkins/workspace/CI_PruebasActualizacion/JmeterPruebas/Empleado2.jmx
Starting standalone test @ Tue Aug 31 17:25:32 PYT 2021 (1630445132626)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
              1 in 00:00:00 = 4,9/s Avg:
29 in 00:00:02 = 17,3/s Avg:
30 in 00:00:02 = 16,0/s Avg:
                                                         40 Min:
                                                                                                    0 (0,00%) Active: 1 Started: 2 Finished: 1
summary +
                                                                       40 Max: 40 Err:
                                                                       3 Max:
summary +
                                                          9 Min:
10 Min:
                                                                                      24 Err:
                                                                                                     0 (0,00%) Active: 0 Started: 20 Finished: 20
                                                                          3 Max:
summary =
                                                                                      40 Err:
                                                                                                     0 (0,00%)
Tidying up ... @ Tue Aug 31 17:25:34 PYT 2021 (1630445134851)
... end of run
+ cd /home/dani/DockerServicio/lightning/lightning-standalone-example
+ echo *******Analisis de aplicacion contenedor Version Produccion***
******Analisis de aplicacion contenedor Version Produccion******
+ java -jar bin/lightning-standalone-7.0.0.jar verify --xml=src/test/resources/xml/example.xml --imeter-
csv=/home/dani/DockerServicio/ReportesTest/PruebaJmLig.csv
Test name:
                           Failed transactions
Test type:
                           passedTransactionsTest
                           Verify number of passed tests
Number of failed transactions <= 0
Test description:
Expected result:
Actual result:
                           Number of failed transactions = 0
Transaction count:
                           60
Test result:
                           Pass
```

====== EXECUTION SUMMARY ==========

Tests executed: 1 Tests passed: 1 Tests failed: 0 Tests errors: 0 Test set status: Pass Execution time: 55ms ##teamcity[buildStatisticValue key='Failed transactions' value='0']

```
+ echo ******Analisis de aplicacion contenedor Version latest******
     ******Analisis de aplicacion contenedor Version latest****
    + java -jar bin/lightning-standalone-7.0.0.jar verify --xml=src/test/resources/xml/example.xml --jmeter-
    csv=/home/dani/DockerServicio/ReportesTest/PruebaJmLigl.csv
    Test name:
                                                                                                   Failed transactions
                                                                                                   passedTransactionsTest
    Test type:
     Test description:
                                                                                                    Verify number of passed tests
    Expected result:
                                                                                                   Number of failed transactions <= 0
    Actual result:
                                                                                                   Number of failed transactions = 0
    Transaction count:
                                                                                                   60
    Test result:
                                                                                                   Pass
           ======== EXECUTION SUMMARY ==========
    Tests executed: 1
    Tests passed:
                                                                                      1
     Tests failed:
                                                                                      0
     Tests errors:
                                                                                      0
    Test set status: Pass
    Execution time:
                                                                                     24ms
    ##teamcity[buildStatisticValue key='Failed transactions' value='0']
    + echo *******ANALITSTS DE VUNERABTI TDADES*******
     *******ANALISIS DE VUNERABILIDADES****
    + echo *******Borrando cache******
    *******Borrando cache*******
    + trivy --clear-cache ci_pruebasactualizacion_app1
    Particle and a state of the state of th
     ******Exportando vulnerabilidades en ReporteAppl
    + trivy --output ReporteApp1 ci_pruebasactualizacion_app1
    2021-08-31T17:25:41.641-0400 [[34mINFO][0m Need to update DB
    2021-08-31T17:25:41.641-0400
                                                                                                                                          [34mINFO][0m Downloading DB...
                                                                                                                                                                                                                           _____] 0.73% ? p/s ?402.23 KiB / 23.12 MiB [-
____] 1.70% ? p/s ?690.32 KiB / 23.12 MiB [-
    172.14 KiB / 23.12 MiB [>
                                                                                                                                                                                                                                                                         _] 2.92% ? p/s ?1010.27 KiB / 23.12 MiB [--
                                                                                                                                                                                                   ] 4.27% 1.36 MiB p/s ETA 16s1.31 MiB / 23.12 MiB [--
                                                                                                                                                                                                 ] 5.65% 1.36 MiB p/s ETA 15s1.55 MiB / 23.12 MiB [--
                                                                                                                                                                                         ] 6.70% 1.36 MIB p/s ETA 15s1.72 MIB / 23.12 MIB [---
] 7.44% 1.36 MIB p/s ETA 15s1.96 MIB / 23.12 MIB [----
] 8.46% 1.36 MIB p/s ETA 15s2.24 MIB / 23.12 MIB [----

        ] 9.4403 1.36 MiB p/s ETA 1552.55 MiB / 23.12 MiB [-----

        ] 9.67% 1.36 MiB p/s ETA 1552.75 MiB / 23.12 MiB [-----

        ] 11.03% 1.36 MiB p/s ETA 1552.74 MiB / 23.12 MiB [------

        ] 11.84% 1.36 MiB p/s ETA 1552.91 MiB / 23.12 MiB [------

        ] 12.58% 1.36 MiB p/s ETA 14s3.08 MiB / 23.12 MiB [------

      j
      12.58% 1.3b M1B p/s ETA 1933.00 .... _
      j
      14.20% 1.33 M1B p/s ETA 1453.37 M1D / 2014

      13.34% 1.33 M1B p/s ETA 1553.28 M1B / 23.12 M1B [------> ______]
      14.20% 1.33 M1B p/s ETA 1453.37 M1D / 2014
      100 / 2014

      M1B [------> _____]
      14.20% 1.33 M1B p/s ETA 1453.52 M1B / 23.12 M1B [------> _____]
      14.20% 1.33 M1B p/s ETA 1453.57 M1D / 2014

      Start
      13.34% 1.32 M1B p/s ETA 153.28 M1B / 23.12 M1B [------> _____]
      15.23% 1.29 M1B p/s ETA 1553.79 M1B / 23.12 M1B [------> _____]
      16.40% 1.29 M1B p/s ETA 1553.96 M1B / 23.12 M1B [------> _____]

      M1R [------> _____]
      17.14% 1.25 M1B p/s ETA 1554.09 M1B / 23.12 M1B [------> ______]
      16.40% 1.29 M1B p/s ETA 1553.96 M1B / 23.12 M1B [-------> ______]

      15.87% 1.29 MiB p/s ETA 1553./9 MID / 25.42 MID

      11/.1% 1/4% 1/4% 1/4% 1/4%
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1
      1

        j
        20.93% 1.22 Has product to the product 
                                                                                                                                                                                                                                                                                                                                                                                                     ] 22.45% 1.21 MiB p/s ETA 14s5.35 MiB / 23.12

      24.81% 1.19 MiB p/s EIA 1455.25 PILD / 25.12 / PIL
> _____] 35.30% 1.00 110 p, c = _____] 36.22% 1.00 100 p/s cm ______]

1.08 MIB p/s ETA 13s8.37 MIB / 23.12 MIB [--------_____] 36.79% 1.05 MIB p/s ETA 13s8.64 MIB / 23.12 MIB [----------____]

______] 36.79% 1.05 MIB p/s ETA 13s8.64 MIB / 23.12 MIB [-----------____]
  1 37.98% 1.03
```

 

 23.12 FLOD [

 -\_\_\_\_\_\_] 46.00% 968.91 KIB p/5 ETA 13510.77 HLB / 2012

 p/s ETA 13s10.82 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_] 47.29% 952.10 KIB p/s ETA 13s11.04 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_] 47.29% 952.10 KIB p/s ETA 13s11.04 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_] 48.30% 929.25 KIB p/s ETA 13s11.30 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_] 48.88% 929.25 KIB p/s ETA 13s11.44 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_] 49.49% 929.25 KIB p/s ETA 13s11.30 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_] 19.47% 909.27 KIB p/s ETA 12s11.53 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_\_] 50.77% 909.27 KIB p/s ETA 12s11.90 MIB / 23.12 MIB [

 -\_\_\_\_\_\_\_\_\_\_\_] 50.77% 909.27 KIB p/s ETA 12s11.90 MIB / 23.12 MIB [

 1
 49.49%
 929.25
 KiB
 p/s
 ETA
 1/2511.23
 FILD
 20.77%
 909.27
 KiB
 p/s
 ETA
 1/2511.33
 FILD
 20.77%
 909.27
 KiB
 p/s
 ETA
 1/2511.34%
 1/2511.34%
 1/2511.34%
 1/2512.35
 MiB
 23.12
 MiB
 23.12 903.33 KIB p/s ETA 12s12.01 MIB / 23.12 MIB [--------] 53.43% 903.33 KIB p/s ETA 12s12.48 MIB / 23.12 MIB [--------] 53.43% 903.33 KIB p/s ETA 12s12.48 MIB / 23.12 MIB [--------] 54.75% 896.83 KIB ] 55.66% 881.74 KiB p/s ETA 11s13.00 MiB / 23.12 MiB [----->\_\_\_\_] 56.21 3.12 MiB [----->\_\_\_\_] 56.78% 881.74 KiB p/s ETA 11s13.20 MiB / 23.12 MiB 1 56.21% 840.43 FLD #/3 

 b21.35 KLD p/5 ] 74.78% 847.50 KiB p/s ETA

7s17.45 MiB / 23.12 MiB [-----> 

 Image: Characterized of the control \_\_\_\_] 01.34% 828.00 KiB p/s ETA 5s19.07 MiB / 23.12 MiB [------] 83.12% 820.05 

 5518.93 MiB / 23.12 MiB [----->\_\_\_\_] \$1.00% 020.00 MiD p,5 -- 

 5518.93 MiB / 23.12 MiB [----->\_\_\_\_] 82.49% 828.00 KiB p/s ETA 5s19.22 MiB / 23.12 MiB [----->\_\_\_\_] 83.62% 820.05 KiB p/s ETA 4s19.48 MiB / 23.12 MiB [----->\_\_\_\_] 83.62% 820.05 KiB p/s ETA 4s19.48 MiB / 23.12 MiB [----->\_\_\_\_]

 1 89,48% 807,21 ->1 100 00% 77/ 07 KiB p/s FTA [[33mWARN[[0m This OS version is not

on the EOL list: alpine 3.14 2021-08-31T17:27:04.629-0400 2021-08-31T17:27:04.650-0400 2021-08-31T17:27:04.696-0400 2021-08-31T17:27:04.696-0400

[][34mINF0□[0m Detecting Alpine vulnerabilities..

∏[33mWARN∏[0m

[[34mINFO][0m Detecting jar vulnerabilities... This OS version is no longer supported by the distribution: alpine 3.14.0

[][33mWARND[[0m ] The vulnerability detection may be insufficient because security updates are not provided

```
ci_pruebasactualizacion_appl (alpine 3.14.0)
Total: 5 (UNKNOWN: 0, LOW: 0, MEDIUM: 2, HIGH: 2, CRITICAL: 1)
usr/src/servicio/rest-0.0.1-SNAPSHOT.jar
Total: 10 (UNKNOWN: 0, LOW: 0, MEDIUM: 3, HIGH: 6, CRITICAL: 1)
+ echo *******Analiza vulnerabilidades*******
   *****Analiza vulnerabilidades****
+ trivy image --light --exit-code 0 --severity (RITICAL ci_pruebasactualizacion_appl
2021-08-31T17:27:04.794-0400 [[34mINFO][0m Need to update DB
2021-08-31T17:27:04.794-0400 [[34mINFO][0m Downloading DB...
192.00 KiB / 5.75 MiB [-->____
                                     ] 3.26% ? p/s ?560.57 KiB / 5.75 MiB [-----
] 9.53% ? p/s ?1.02 MiB / 5.75 MiB [------
.
_] 27.81% 1.70 MiB p/s ETA 2s1.86 MiB / 5.75
1 54.73% 1.64 MiB
----->] 100.00% 1.37 MiB p/s ETA 0s5.75 MiB / 5.75 MiB [------->] 100.00% 1.37 MiB p/s ETA 0s5.75 MiB / 5.75 MiB [--------->] 100.00% 1.37 MiB p/s ETA 0s5.75 MiB / 5.75 MiB [-------->]
MIB / 5.75 MIB [----->] 100.00% 1.28 MIB p/s ETA 0s5.75 MIB / 5.75 MIB [----
                                                                            ---->1 100.00% 1.28 MiB p/s
----->] 100.00% 1.20 MiB p/s ETA 0s5.75 MiB / 5.75 MiB [------
                                                                                    -->1 100.00%
----->] 100.00% 1.20 MiB p/s ETA 0s5.75 MiB / 5.75 MiB [----
----->] 100.00% 1.12 MiB p/s ETA 0s5.75 MiB /
5.75 MiB [-----
  .....] 100.00% 737.29 KiB p/s 8s2021-08-31T17:27:14.258-0400
                                                   [[33mWARN][0m This OS version is not on the EOL list: alpine
3.14
3.14
2021-08-31T17:27:14.258-0400 []34mINFO_[0m Detecting Alpine vulnerabilities...
2021-08-31T17:27:14.297-0400 []3mWARN[][0m This OS version is no longer supported by the distribution: alpine 3.14.0
[]33mWARN[][0m This OS version is no longer supported by the distribution: alpine 3.14.0
2021-08-31T17:27:14.332-0400 [[33mWARN[[0m The vulnerability detection may be insufficient because security updates are not provided
ci pruebasactualizacion appl (alpine 3.14.0)
Total: 1 (CRITICAL: 1)
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |
| apk-tools | CVE-2021-36159 | CRITICAL | 2.12.5-r1 |
                                       2.12.6-r0
usr/src/servicio/rest-0.0.1-SNAPSHOT.jar
Total: 1 (CRITICAL: 1)
              | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |
     LIBRARY
| net.minidev:json-smart | CVE-2021-27568 | CRITICAL |
                                            2.3 | 2.4.1, 1.3.2 |
                         Global Slack Notifier try posting to slack. However some error occurred
TeamDomain :tesis-sow3593
Channel :proyecto
Message :Ejecución Exitosa
Lanzando una nueva ejecución de CI_PruebasDespliegue
Finished: SUCCESS
```

Figura 91 Log de Jenkins al ejecutar la tarea CI\_PruebasActualizacion.

En la figura 91 se observa el proceso completo de ejecución de la tarea CI\_PruebasActualizacion completando así la construcción de la tarea.

## 4.1 Formas de implementar la tarea de actualización

La actualización de la infraestructura de los servicios pueden ser establecidos por diferentes criterios se cita algunos de ellos:

**Política de implementación:** En este caso el equipo de seguridad establece un periodo de tiempo para la actualización del SO de los contenedores, para ello se dicta unos criterios a tener en cuenta:

La actualización de los contenedores será clasificada por orden de importancia y criticidad dentro de la organización, siendo estos los primeros en actualizarse..

Solo se actualizarán los SO de los contenedores, en caso de actualización de una plataforma específica de desarrollo, se actualizará con la aprobación del departamento de desarrollo.

Los Dockerfile con las versiones actualizadas deben ser depositados en un repositorio exclusivo para actualizaciones, clasificado por plataformas de desarrollo y deberán ser creados por las personas del equipo de desarrollo con la asesoría del equipo de seguridad y aprobado por el equipo de operaciones.

Los tester deberán realizar pruebas de estrés y funcionalidad sobre los servicios expuestos en la plataforma actualizada.

En caso de un funcionamiento incorrecto en la actualización, se informará al equipo de desarrollo, el cual coordinará para su revisión con el equipo de seguridad y de acuerdo al resultado del análisis realizado entre estos dos departamentos se evaluará la actualización o no del contenedor.

**Por vulnerabilidades encontradas:** En este escenario los contenedores de los servicios que sean dispuestos para actualizar pasarán por un escáner en busca de vulnerabilidades críticas, en caso de hallar una vulnerabilidad de nivel crítico se decidirá su actualización de acuerdo al informe generado de las vulnerabilidades analizadas por el equipo de seguridad, desarrollo y QA, en este caso se podría detener la CI y someter a evaluación la actualizacion.

Si no se encuentra ninguna falla crítica, la actualización del mismo será tomada como una recomendación y quedará a criterio del equipo de seguridad y QA actualizar el contenedor.

## 4.2 Ventajas de la actualización continua

Permite anticipar el resultado de una implementación de un servicio en un contenedor actualizado desde el aspecto funcional y de seguridad.

Permite la utilización de un directorio común para las pruebas de los servicios, debido al docker compose solo se debe actualizar el Dockerfile y el servicio para realizar la prueba.

Emite reportes de las vulnerabilidades que pueden encontrarse o de las pruebas funcionales que puedan fallar, en caso de querer utilizar esta información como apoyo adicional para la toma de decisiones en la actualización del componente.

La actualización de los contenedores reduce las vulnerabilidades que puedan explotarse por desactualización.

Debido a que la actualización se realiza de forma frecuente, se evita la inversión de un tiempo y recurso para este fin.

## 5. Conclusión

A lo largo del presente trabajo se presentaron diferentes herramientas para la implementación de una tarea dentro de un flujo de una CI, encargada de determinar si un servicio puede ser implementado en una plataforma actualizada.

El objetivo general de este trabajo que se planteó al comienzo del mismo es: "Presentar un esquema de actualización continua para servicios web Spring Boot en contenedores docker como un componente más dentro de una CI, de forma a mitigar los problemas de seguridad dado por los componentes obsoletos que puedan existir"; por su parte la hipótesis en que se inscribe este trabajo es: "Se puede integrar pruebas de actualización sobre las imágenes de los SO utilizando las herramientas de Jmeter, Lightning y Trivy para servicios web Spring boot desplegados dentro de contenedores Docker, integrando al equipo de seguridad, QA y operaciones".

Existen una gran variedad de formas de integrar una tarea de actualización dentro de una CI, sin embargo no existe un enfoque definido para realizar las actualizaciones de las plataformas de infraestructura como parte de un flujo en una CI. Con la idea de cumplir el objetivo del trabajo buscando un enfoque innovador se planteó a lo largo del mismo un conjunto de metodologías y herramientas que permiten encarar esta idea desde una perspectiva de integración e interacción entre los departamentos de Desarrollo, QA, Seguridad y Operaciones.

Para lograr el objetivo se propuso una metodología Top-Down que va desde lo más general a lo particular, es decir, inicia desde la relación entre los departamentos de la organización hasta la aplicación de las herramientas en cada uno de ellas finalizando en una integración tanto organizacional como técnica.

Este enfoque Top-Down se divide a su vez en dos partes bien diferenciadas y permite validar la hipótesis, la primera parte inicia cuando se describen los departamentos que necesitan involucrarse para la coordinación de las tareas y que interacción debe haber entre ellas y la segunda es que tareas técnicas deben realizar cada departamento con las herramientas mencionadas Jenkins, Trivy, Apache Jmeter, Lightning, GitHub para así poder realizar las pruebas de actualización del servicio dentro de una CI. Sin entrar en mucho detalle se puede afirmar que estos dos enfoques están basados en las respuestas que el servicio pueda responder en una plataforma actualizada y si esta plataforma no posee vulnerabilidades críticas. En síntesis puede decirse que este trabajo presenta un esquema de actualización para servicios basados en el framework Spring Boot sobre la plataforma Docker como parte de una CI.

## 6. Bibliografía

[1] Baeldung, «Learn Spring Boot», 16 Junio 2021.

[En línea].Disponible:https://www.baeldung.com/spring-boot

[2] Oracle, «Oracle»,16 Junio 2021.

[En línea].

Disponible:

https://www.oracle.com/java/technologies/javase-jdk11-downloads.html

[3] NetBeans, «NetBeans»,16 Junio 2021.[En línea].

Disponible:https://netbeans.apache.org/download/nb123/nb123.html

[4] Spring initializr, «Spring Boot», 16 Junio 2021.

[En línea]. Disponible: https://start.spring.io/

[5] Building REST Services with Spring, «Git Hub», ,16 Junio 2021.

Disponible:https://github.com/spring-guides/tut-rest

[6] Docker, «Docker, »12 Agosto 2020.

[En línea]. Disponible:https://www.docker.com/resources/what-container

[7] Docker Docs, «Docker,»12 Agosto 2020.

[En línea]. Disponible:https://docs.docker.com/engine/install/ubuntu/

[8] «aquasecurity/trivy».10 Febrero 2019[En línea].

Disponible: https://github.com/aquasecurity/trivy.git.

[9] Trivy, «Trivy».10 Febrero 2019[En línea]

,Disponible:https://aquasecurity.github.io/trivy/v0.19.1/getting-started/installati on/

[10] Docker Docs, «Docker,»12 Agosto 2020.

[En línea].

Disponible:https://docs.docker.com/engine/reference/builder/#:~:text=A%20D ockerfile%20is%20a%20text,command%2Dline%20instructions%20in%20su ccession.

[11] Wikipedia, «Jenkins,»12 Agosto 2020.

[En línea]. Disponible: https://es.wikipedia.org/wiki/Jenkins
[12]Jenkins, «Jenkins, »Mayo 2021 [En Línea],
Disponible: https://www.jenkins.io/
[13] «lightning, »10 Junio 2020[En línea].
Available: https://automatictester.github.io/lightning/
[14] «lightning test type».10 Junio 2020.[En línea]
Available: https://automatictester.github.io/lightning/test\_types.html
http://automatictester.github.io/lightning/standalone\_jar.html
[15] JMeter, «JMeter,, »Mayo 2021 [En Línea],
Disponible: https://jmeter.apache.org/
[16] AlphineLinux, «AlphineLinux, »11 Agosto 2020.
[En línea]. Available: https://alpinelinux.org/about/