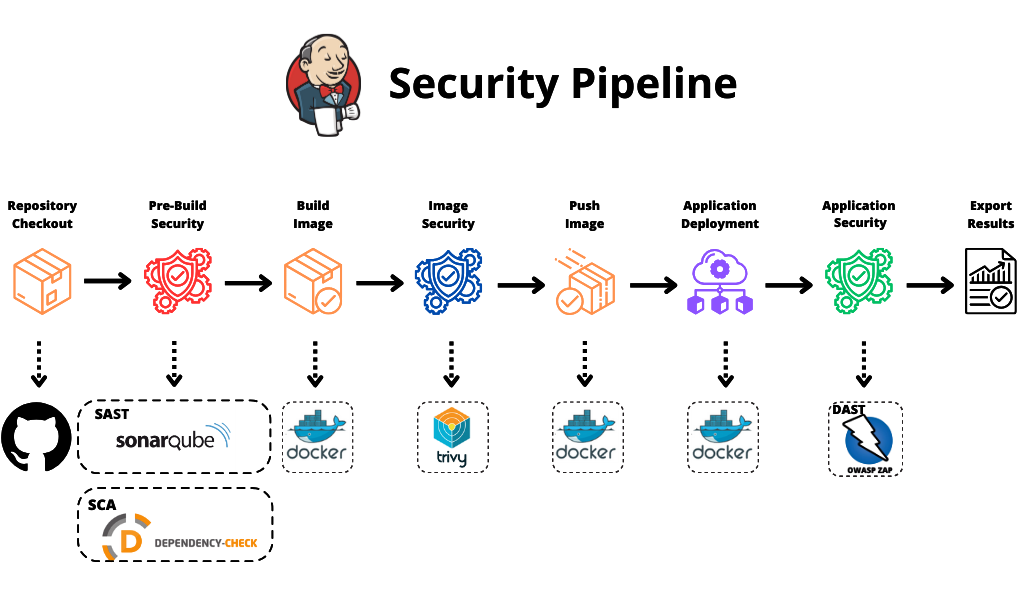
**FELIPE ANDRÉS CRUZ VERDUGO**

**Pipeline CI/CD en Jenkins**

**Marzo - 2023**

**INTRODUCCIÓN**

El presente informe tiene como objetivo implementar un pipeline de Integración Continua. Para ello se considerará el siguiente flujo de trabajo:

****

Como se aprecia en la imagen anterior, las etapas en el pipeline consisten en:

1. Clonación del repositorio desde GitHub.
2. Análisis SAST con Sonarqube.
3. Análisis SCA con Dependency-Check.
4. Construcción de la Imagen Docker.
5. Análisis de Imagen Docker con Trivy.
6. Despliegue de la aplicación con Docker.
7. Análisis DAST con Owasp ZAP.

Adicionalmente se tiene la condición para los análisis SAST, SCA e Image Security que en caso de encontrar vulnerabilidades de categoría **media, alta o crítica** el pipeline debe fallar.

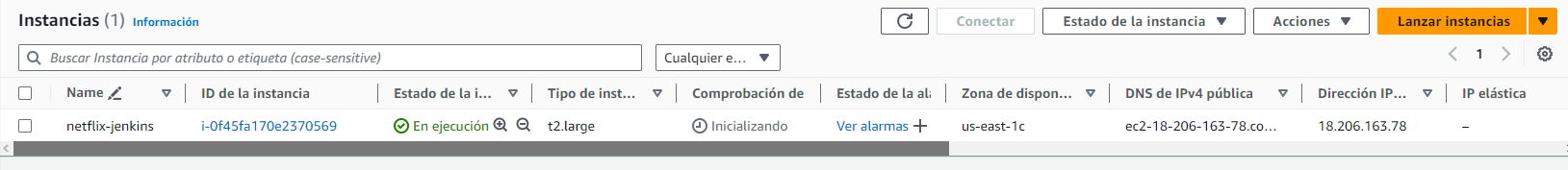
El documento se estructura de la siguiente manera:

* Prerrequisitos: Configuración del ambiente.
* Desarrollo:
  + Clonación de repositorio.
  + Creación pipeline.
  + Resultados.
  + Aplicativo.

**Prerrequisitos**

**Levantar EC2**

* Levantar EC2 tipo t2.large con la IP elástica (Para que no vaya cambiando):



* Habilitar los puertos 8080, 8081 y 9000:

Interfaz de usuario gráfica

Descripción generada automáticamente

* Conectarse a EC2 mediante SSH desde la terminal abierta en la ubicación de la clave pem:

ssh -i EC2-devsecops-project.pem ubuntu@44.194.185.232

**Instalar jq**

sudo apt-get update

sudo apt-get install jq -y

**Instalar Java**

sudo apt update

sudo apt install fontconfig openjdk-17-jre

java -version

openjdk version "17.0.8" 2023-07-18

OpenJDK Runtime Environment (build 17.0.8+7-Debian-1deb12u1)

OpenJDK 64-Bit Server VM (build 17.0.8+7-Debian-1deb12u1, mixed mode, sharing)

**Instalar Docker**

sudo apt-get update

sudo apt-get install docker.io -y

sudo usermod -aG docker ubuntu

newgrp docker

sudo chmod 777 /var/run/docker.sock

**Instalar Sonarqube**

docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

Ingresar a Sonarque: http://54.144.159.54:9000

Por defecto las credenciales son admin-admin y se solicitará cambiar la password.

sonarqube123

Generar token: Administration/Security/Tokens: squ\_afac437b0f169492bddb1e8295c0026e771af93e

**Instalar Trivy**

sudo apt-get install wget apt-transport-https gnupg lsb-release

wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -

echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb\_release -sc) main | sudo tee -a /etc/apt/sources.list.d/trivy.list

sudo apt-get update

sudo apt-get install trivy

**Instalar Jenkins**

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins

sudo systemctl start jenkins

sudo systemctl enable Jenkins

Revisar estado de Jenkins: sudo systemctl status jenkins

Reiniciar Jenkins: sudo systemctl restart jenkins

Detener Jenkins: sudo systemctl stop jenkins

Ingresar a Jenkins: http://54.144.159.54:8080

Para conocer la password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

jenkins123

**Descargar Plugins de Jenkins:**

* Eclipse Temurin Installer: Para el JDK
* SonarQube Scanner
* NodeJs Plugin
* OWASP Dependency-Check
* Docker:
  + Docker
  + Docker Commons
  + Docker Pipeline
  + Docker API
  + Docker-build-step
* Email Extension Plugin

**Obtener API Key de National Vulnerability Database (NTV)**

<https://nvd.nist.gov/developers/request-an-api-key>

53c53bc6-971a-4d13-be42-b5673aa6364b

**Obtener API Key de TMDB**

<https://www.themoviedb.org/settings/api>

a39af0296e3f125c9e57ba803453c93a

**Configurar tokens en Jenkins**

Manage Jenkins/Credentials/global/Add Credentials

* Add Sonarqube token como Secret text: Para realizar escaner.
* Add Sonarqube token como username with password: Para la API.
* Add DockerHub token con Unsername with password
* Add Dependency-check token como Secret text.
* Add TMBD API Key como Secret text.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Configuración de herramientas**

* Sonarqube:

Manage Jenkins/System/SonarQube servers:

* Name: sonar-server
* URL: URL donde está levantado sonar. http://54.144.159.54:9000
* Agregar token creado.

Manage Jenkins/Tools/Instalaciones de Sonarqube Scanner:

* Name: sonar-scanner
* Versión: Seleccionar la última desde Maven Central
* JDK:

Manage Jenkins/Tools/Instalaciones de JDK:

* Nombre: jdk17
* Instalar automáticamente from adoptium.net: jdk-17.0.8.1+1
* NodeJs:

Manage Jenkins/Tools/Instalaciones de NodeJS:

* Nombre: node16
* Versión: NodeJS 16.2.0
* Docker:

Manage Jenkins/Tools/Instalaciones de NodeJS:

* Nombre: docker
* Instalar automáticamente from docker.com: latest
* Dependency-Check:

Manage Jenkins/Tools/Instalaciones de Dependency-Check:

* Nombre: DP-Check
* Instalar automáticamente from github.com: dependency-check 9.0.9

**Conexión entre Jenkins y GitHub para despliegue automático del pipeline**

GitHub:

* Settings/Developer Settings/Personal Access tokens/Tokens (classic)/Generate token

ghp\_4D1KMfqM8c5jxTYwnJ4AeKLGsEI8zq1rOlxe

* Repositorio/Settings/Webhooks/Add webhook

Payload URL: URL de Jenkins: http://54.144.159.54:8080/github-webhook/

Con esto se indica que cada commit sea enviado al Jenkins.

Captura de pantalla de un celular

Descripción generada automáticamente

**DESARROLLO**

**Clonación de repositorio**

Clonación de repositorio:

El repositorio original es: <https://github.com/DevSecOps-Ciclo-7/final-lab.git>

Fork del repositorio: <https://github.com/Pipe-Cruz/final-lab.git>

La rama creada para la implementación del pipeline se llama ***Jenkins-CICD:***

Captura de pantalla de un celular

Descripción generada automáticamente

**Creación de pipeline**

Dentro del repositorio se debe crear un archivo Jenkinsfile:

**Texto

Descripción generada automáticamente**

Posteriormente, en Jenkins se debe crear el pipeline: Nueva Tarea/Pipeline/

* Name: netflix
* Habilitar la casilla *GitHub hook triger for GITScm polling*
* Definition/Pipeline script from SCM
* SCM: Git
* Repository URL: https://<GITHUB TOKEN>@github.com/Pipe-Cruz/final-lab.git
* Branch: Jenkins-CICD
* Script Path: Jenkinsfile

De este modo, cada vez que se realice un commit a la rama Jenkins-CICD se ejecutará el pipeline de manera automatizada.

Si el pipeline falla por la conexión a Docker, utilizar los siguientes comandos en la instancia o donde se encuentre levantado Jenkins:

sudo su

sudo usermod -aG docker jenkins

**Resultados**

En la siguiente imagen se puede observar la ejecución del pipeline:

Aplicación, Tabla

Descripción generada automáticamente

**Logs**

A partir de la ejecución del pipeline se pueden extraer los siguientes logs:



**Reportes**

* SAST: SonarQube

En el pipeline Sonarqube no entrega un reporte, pero al finalizar la ejecución se puede visualizar el SonarQube Quality Gate con dos parámetros:

- Succes: Indica que el análisis se ha ejecutado correctamente.

- Passed: Indica que el código ha pasado los parámetros definidos por defecto en sonarqube.

Dado que SonarQube se encuentra levantado en el puerto 9000, podemos acceder a él para ver mediante interfaz los resultados:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto, Sitio web

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**Condición**

Sonarqube no utiliza las categorías Media, Alta y Crítica, sino Major, Critical y Blocker, en base a eso se configuró una condición que al identificar alguna vulnerabilidad dentro de estas categorías el pipeline se detenga:

Tabla, Calendario

Descripción generada automáticamente

Como se puede apreciar el pipeline falla en el stage SonarQube Scan, y en los logs se visualiza el siguiente mensaje configurado:



* SCA: Dependency-Check

En los Artifacts se puede visualizar el reporte en formato XML:



Adicionalmente Jenkins muestra de forma gráfica los resultados obtenidos:

Gráfico

Descripción generada automáticamente con confianza media

Tabla

Descripción generada automáticamente con confianza media

Como se puede apreciar, se han encontrado tres vulnerabilidades, 1 de categoría media y dos de categoría alta.

**Condición**

Activando la condición de que en caso de encontrar vulnerabilidades medias, altas o críticas se detenga se tiene lo siguiente:

Tabla

Descripción generada automáticamente

Como se puede apreciar el pipeline falla en el stage Dependency-Check Scan, y en los logs se visualiza el siguiente mensaje configurado:



* Image Security: Trivy

En los Artifacts se puede visualizar el reporte en formato JSON:



Texto

Descripción generada automáticamente

A partir del CWE se puede buscar en la [página oficial](https://cwe.mitre.org/index.html) y encontrar más detalles sobre la vulnerabilidad y formas de mitigarla.

**Condición**

Activando la condición de que en caso de encontrar vulnerabilidades medias, altas o críticas se detenga se tiene lo siguiente:

**Tabla, Calendario

Descripción generada automáticamente**

Como se puede apreciar el pipeline falla en el stage Trivy Image Scan, y en los logs se visualiza el siguiente mensaje configurado:

****

* DAST: Owasp zap

En los Artifacts se puede visualizar el reporte en formato HTML:

Texto

Descripción generada automáticamente con confianza baja

Cabe mencionar que el análisis ejecutado fue en la versión [Baseline Scan](https://www.zaproxy.org/docs/docker/baseline-scan/), dado que, si bien existe una versión [Full Scan](https://www.zaproxy.org/docs/docker/full-scan/), esta tarda un tiempo considerablemente mayor, y pensando en un pipeline que se ejecute para cada commit, se contempla como mejor alternativa la versión baseline, aunque esto queda a criterio de la frecuencia con que se realicen los commits.

**Aplicativo**

Si el pipeline se ha ejecutado de manera exitosa el aplicativo se visualizará en:

http://44.194.185.232:8081

Pantalla de computadora de una persona

Descripción generada automáticamente

**Monitoreo**

**Levantar EC2**

* Levantar EC2 tipo t2.medium (requisitos mínimos de Prometheus) con la ip elástica y habilitar el puerto 3000, 9090 y 9100.

**Instalar Prometheus y Grafana**

sudo useradd --system --no-create-home --shell /bin/false prometheus

wget <https://github.com/prometheus/prometheus/releases/download/v2.47.1/prometheus-2.47.1.linux-amd64.tar.gz>

Extraer .zip:

tar -xvf prometheus-2.47.1.linux-amd64.tar.gz

cd prometheus-2.47.1.linux-amd64/

sudo mkdir -p /data /etc/prometheus

sudo mv prometheus promtool /usr/local/bin/

sudo mv consoles/ console\_libraries/ /etc/prometheus/

sudo mv prometheus.yml /etc/prometheus/prometheus.yml

Prometheus es la aplicación y prom es la herramienta para realizar las consultas.

Cambiar permisos:

sudo chown -R prometheus:prometheus /etc/prometheus/ /data/

Crear Servicio de Prometheus:

sudo nano /etc/systemd/system/prometheus.service

Pegar el siguiente contenido:

[Unit]

Description=Prometheus

Wants=network-online.target

After=network-online.target

StartLimitIntervalSec=500

StartLimitBurst=5

[Service]

User=prometheus

Group=prometheus

Type=simple

Restart=on-failure

RestartSec=5s

ExecStart=/usr/local/bin/prometheus \

--config.file=/etc/prometheus/prometheus.yml \

--storage.tsdb.path=/data \

--web.console.templates=/etc/prometheus/consoles \

--web.console.libraries=/etc/prometheus/console\_libraries \

--web.listen-address=0.0.0.0:9090 \

--web.enable-lifecycle

[Install]

WantedBy=multi-user.target

Explicación:

**User** and **Group** specify the Linux user and group under which Prometheus will run.

**ExecStart** is where you specify the Prometheus binary path, the location of the configuration file (prometheus.yml), the storage directory, and other settings.

**web.listen-address** configures Prometheus to listen on all network interfaces on port 9090.

**web.enable-lifecycle** allows for management of Prometheus through API calls.

Iniciar servicio:

sudo systemctl enable prometheus

sudo systemctl start prometheus

Para revisar el estado:

sudo systemctl status prometheus

Ver Prometheus levantado:

IP:9090

<http://54.165.204.254:9090>

**Instalar Node Exporter**

Node Exporter es un agente que recopila métricas del sistema y las expone en un formato que Prometheus puede ingerir.

sudo useradd --system --no-create-home --shell /bin/false node\_exporter

wget <https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz>

Extraer .zip:

tar -xvf node\_exporter-1.6.1.linux-amd64.tar.gz

sudo mv node\_exporter-1.6.1.linux-amd64/node\_exporter /usr/local/bin/

rm -rf node\_exporter\*

Crear Servicio Node Exporter:

sudo nano /etc/systemd/system/node\_exporter.service

Pegar el siguiente contenido:

[Unit]

Description=Node Exporter

Wants=network-online.target

After=network-online.target

StartLimitIntervalSec=500

StartLimitBurst=5

[Service]

User=node\_exporter

Group=node\_exporter

Type=simple

Restart=on-failure

RestartSec=5s

ExecStart=/usr/local/bin/node\_exporter --collector.logind

[Install]

WantedBy=multi-user.target

Iniciar Servicio:

sudo systemctl enable node\_exporter

sudo systemctl start node\_exporter

Revisar estado del servicio:

sudo systemctl status node\_exporter

**Configurar Prometheus**

Por defecto Prometheus está monitoreando el puerto 9000, debemos cambiar el *endpoint* a nuestra aplicación.

cd /etc/prometheus/

sudo nano prometheus.yml

Agregar las siguientes líneas:

- job\_name: 'node\_exporter'

static\_configs:

- targets: ['localhost:9100']

- job\_name: 'jenkins'

metrics\_path: '/prometheus'

static\_configs:

- targets: ['<your-jenkins-ip>:<your-jenkins-port>']

Para confirmer que la identación del yml es correcta:

promtool check config /etc/prometheus/prometheus.yml

Para reiniciar Prometheus sin reiniciar:

curl -X POST <http://localhost:9090/-/reload>

**Instalar Grafana**

sudo apt-get update

sudo apt-get install -y apt-transport-https software-properties-common

Add GCP Key

wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add –

Add Grafana Repository

echo "deb https://packages.grafana.com/oss/deb stable main" | sudo tee -a /etc/apt/sources.list.d/grafana.list

Actualizar e instalar Grafana

sudo apt-get update

sudo apt-get -y install Grafana

Inicializar servicio

sudo systemctl enable grafana-server

sudo systemctl start grafana-server

Revisar estado del servicio

sudo systemctl status grafana-server

Acceder a la aplicación

<http://54.165.204.254:3000>

admin

admin

grafana123