Introducción a DevSecOps para la mejora de los procesos de desarrollo de software con herramientas Open Source

Trabajo Final del Grado de Ingeniería Informática

Autor: Alberto Élez Villamarín
Tutor: Joaquín López Sánchez-Montañés
Área de Proyecto: Administración de redes y sistemas operativos
Fecha: Junio de 2023

Contenido de la presentación

- 1. Objetivos del TFG
- 2. Introducción a DevOps
- 3. Introducción a DevSecOps
- 4. Aplicación práctica de implementación de DevSecOps
- 5. Conclusiones

1. Objetivos del TFG

Justificación del TFG

Aunque la cultura DevOps está actualmente ampliamente adoptada en las empresas que desarrollan software, todavía no existe una mentalidad de aplicar la seguridad desde el inicio de un proyecto y, por tanto, no se ha estandarizado la implementación de DevSecOps.

Este proyecto pretende explicar como implementar DevSecOps en una empresa de desarrollo de software, proporcionado un conjunto inicial de herramientas Open Source, un manual para instalarlas y un pipeline completo de Jenkins para poderlo incorporar rápidamente a sus pipelines DevOps de Jenkins.

Objetivos del TFG (I)

Se definen cuatro objetivos principales, cada uno de ellos desarrollado en un capítulo de la memoria:

- El primer objetivo que se pretende alcanzar con la realización de este TFG es dar una visión completa de la cultura o el movimiento DevOps: porqué surgió la necesidad de implantarlo en las empresas, sus principios, sus roles, su ciclo de vida, su relación con las metodologías ágiles, sus ventajas e inconvenientes, etc.

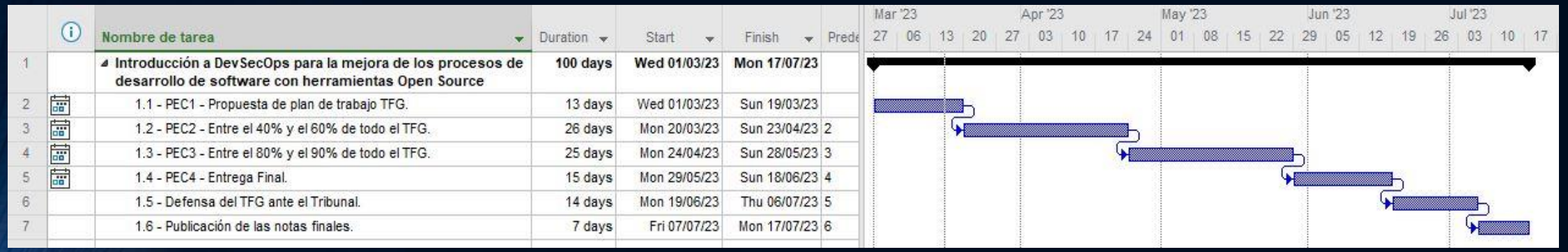
Objetivos del TFG (II)

- El segundo objetivo es analizar el modelo DevSecOps, porqué es necesario implementarlo desde las fases iniciales del proyecto, incluyendo procedimientos de automatización y análisis de código (estático y dinámico) y estableciendo un ciclo de vida de desarrollo de software seguro (SSDLC).
- El tercer objetivo es analizar el estado el arte actual de las herramientas empleadas en DevOps y en DevSecOps, dando una visión de las más importantes para cada una de las áreas involucradas en todas las fases de los proyectos de desarrollo de software.

Objetivos del TFG (III)

- El cuarto objetivo es desarrollar un ejemplo de implantación de un proyecto de desarrollo de software cubriendo todo el ciclo de vida DevSecOps (SSDLC), empleando herramientas Open Source (un pipeline de Jenkins completo), de manera que cualquier empresa pueda utilizarlo, adaptándolo a sus necesidades, y alcance de esa manera un DevSecOps Maturity Model Level 1 (DSOMM Level1).

Planificación del TFG



	Nombre de tarea	Duration	Start	Finish	Prede
1	**Introducción a DevSecOps para la mejora de los procesos de desarrollo de software con herramientas Open Source**	**100 days**	**Wed 01/03/23**	**Mon 17/07/23**	
2	1.1 - PEC1 - Propuesta de plan de trabajo TFG.	13 days	Wed 01/03/23	Sun 19/03/23	
3	1.2 - PEC2 - Entre el 40% y el 60% de todo el TFG.	26 days	Mon 20/03/23	Sun 23/04/23	2
4	1.3 - PEC3 - Entre el 80% y el 90% de todo el TFG.	25 days	Mon 24/04/23	Sun 28/05/23	3
5	1.4 - PEC4 - Entrega Final.	15 days	Mon 29/05/23	Sun 18/06/23	4
6	1.5 - Defensa del TFG ante el Tribunal.	14 days	Mon 19/06/23	Thu 06/07/23	5
7	1.6 - Publicación de las notas finales.	7 days	Fri 07/07/23	Mon 17/07/23	6

2. Introducción a DevOps

¿Qué es DevOps?

El término DevOps surge de la combinación de las palabras “desarrollo” (**Dev**elopment) y “operaciones” (**Op**eration**s**), ya que se trata de un conjunto de principios y prácticas que optimizan el tiempo de entrega de un producto software, automatizan todo lo posible, gestionan la infraestructura como código y mejoran la experiencia del usuario en base a la retroalimentación de todo el proceso.

DevOps no se refiere solo a automatización y herramientas, sino que tiene tres componentes fundamentales: procesos, organización y tecnología.

El objetivo principal de DevOps es acelerar la entrega de valor al cliente.

Conceptos en los que se fundamenta DevOps (I)

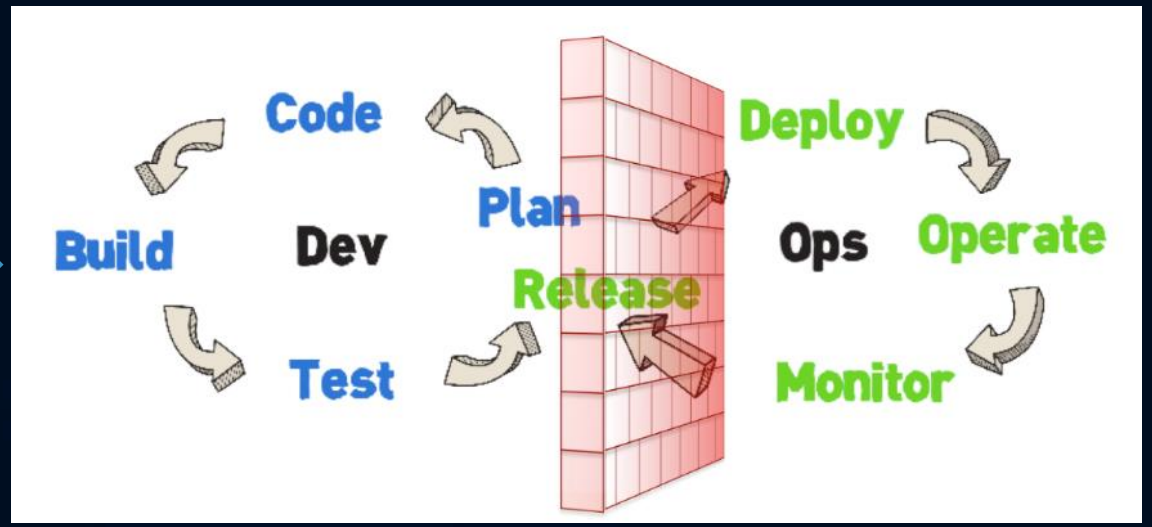
DevOps se fundamenta en los siguientes conceptos principales:

- Mejora Continua: DevOps defiende la mejora continua para minimizar el desperdicio.
- Automatizar todo lo posible: la automatización es un principio central del proceso DevOps. No solo durante la fase de desarrollo del software, sino también para la gestión de la infraestructura. Se deben automatizar todos los procesos manuales repetitivos y que no aporten valor
- Trabajar como un solo equipo: DevOps trata de acabar con los silos y las barreras entre los distintos departamentos. Cada trabajador es responsable de su parte del proyecto.
- Monitorización y pruebas continuas: para detectar los errores lo antes posible y mejorar la calidad de los productos generados.

Conceptos en los que se fundamenta DevOps (II)

DevOps es ante todo un cambio de mentalidad, acaba con los silos dentro de las organizaciones y fomenta el trabajo colaborativo:

DevOps



Mejoras que proporciona DevOps

- Mejora de la frecuencia de despliegue: en el mundo del desarrollo de software tradicional se suelen compilar muchas versiones, pero se despliegan muy pocas. Uno de los objetivos que se cumplen utilizando metodología DevOps es mejorar la frecuencia y la calidad de los despliegues. Para ello se utilizan herramientas como el Continuous Delivery (entrega continua) y el Continuous Deployment (despliegue continuo).

- Mejora del Time to Market: al utilizar las herramientas anteriores para mejorar la frecuencia de despliegue permite salir al mercado de manera más ágil, detectar los errores antes y empezar a trabajar con productos terminados más rápidamente.

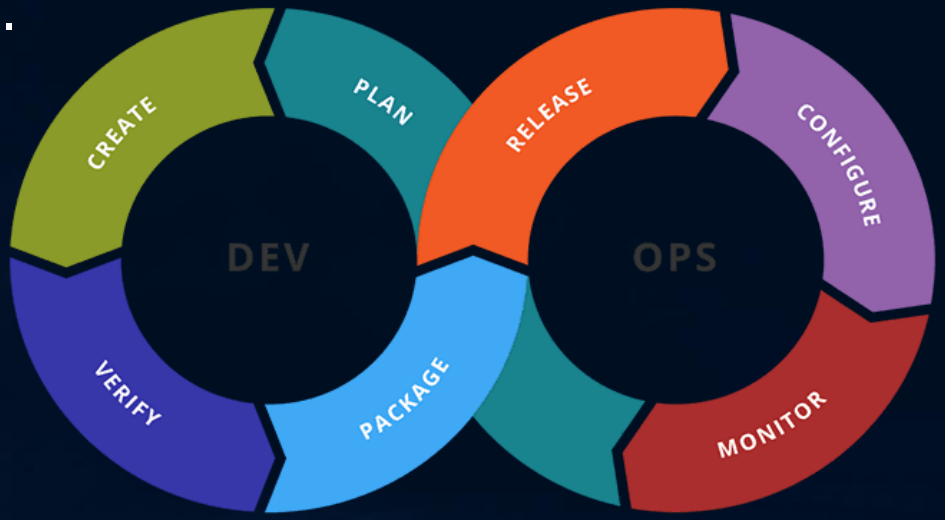
- Reducción de la tasa de error: la aplicación de ciclos de monitorización, testeo continuo y corrección de errores detectados, permiten detectar los errores que pasan al entorno de producción y corregirlos produciendo el menor impacto posible y reduciendo así la tasa y los tiempos de error de los productos.

- Reducción de los tiempos de corrección: al integrar de manera continua, se trabaja con menos cantidad de cambios en cada delivery y al emplear automatización y monitorización continuas, las correcciones se realizarán en un tiempo más corto al encontrar con mayor facilidad el origen de los fallos.

Ciclo de vida de DevOps (I)

Se basa en el ciclo de vida de las metodologías ágiles y es por tanto iterativo.

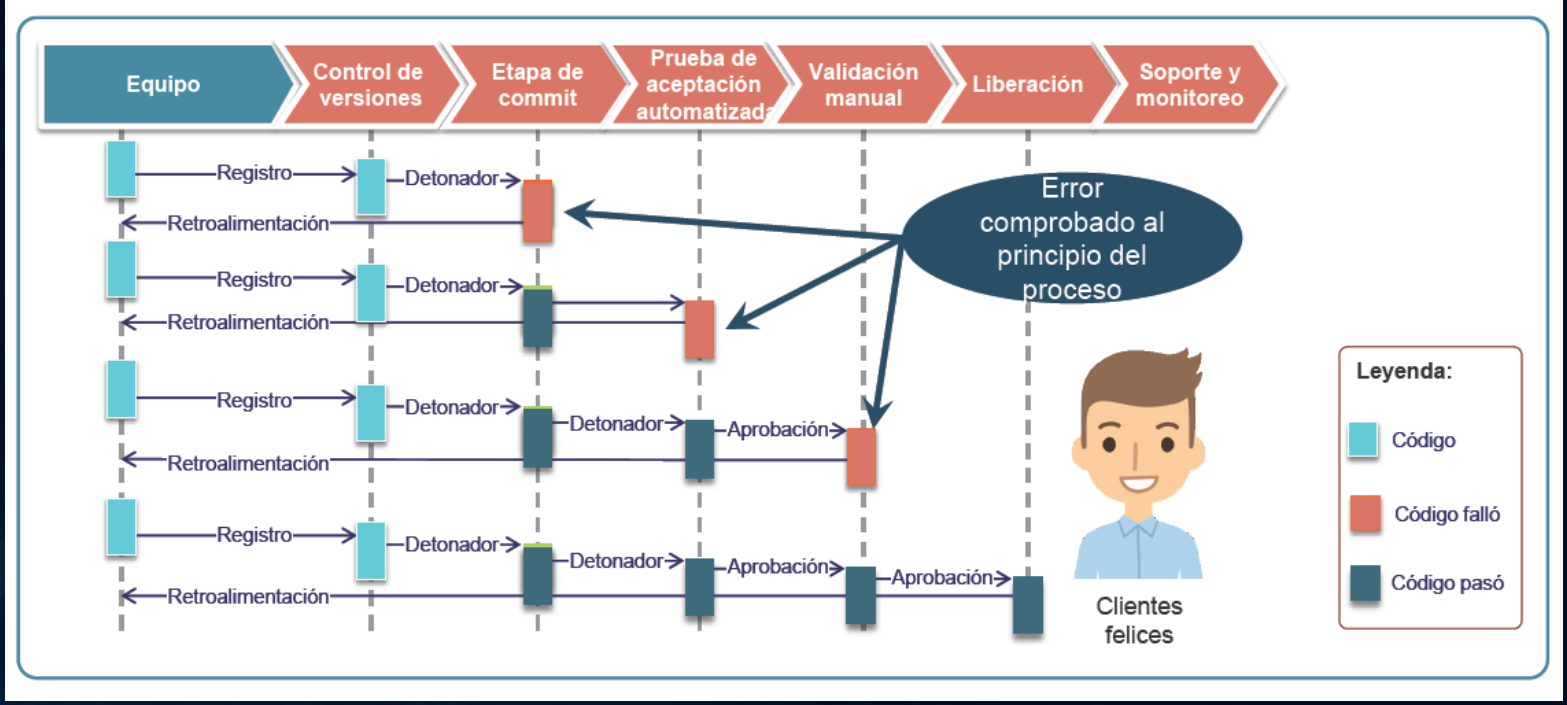
Las fases que lo integran forman un bucle infinito donde entran en juego las tareas propias de dos mundos: el de Desarrollo y el de Operaciones.



Fuente: https://www.uxland.es/devops-integracion-continua/

Ciclo de vida de DevOps (II)

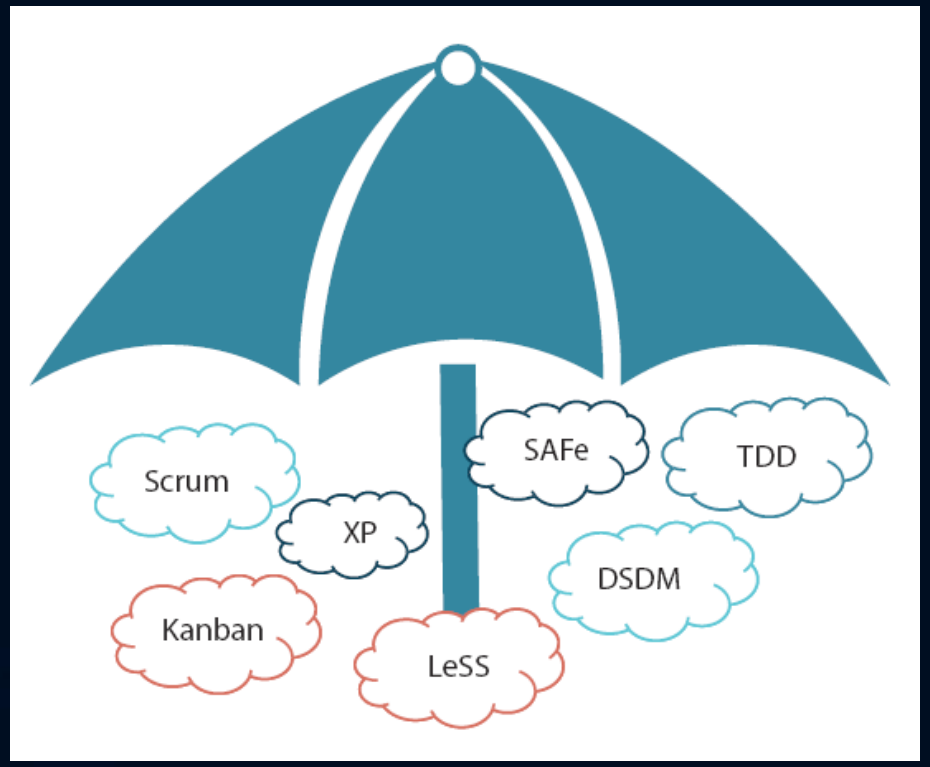
Lo que pretende conseguir el ciclo de vida DevOps es detectar los errores lo antes posible, utilizando procesos automatizados, para entregar valor al cliente lo antes posible.



Fuente: https://www.devopsagileskills.org/

Metodologías Ágiles

El ciclo de vida en DevOps se basa en el ciclo de vida de las metodologías ágiles y es por tanto iterativo, compuesto por diferentes fases que buscan siempre el incremento de valor y la mejora continua, automatizando todo lo que sea posible. Para ello se basa en otros frameworks de metodologías ágiles.



Fuente: https://www.devopsagileskills.org/

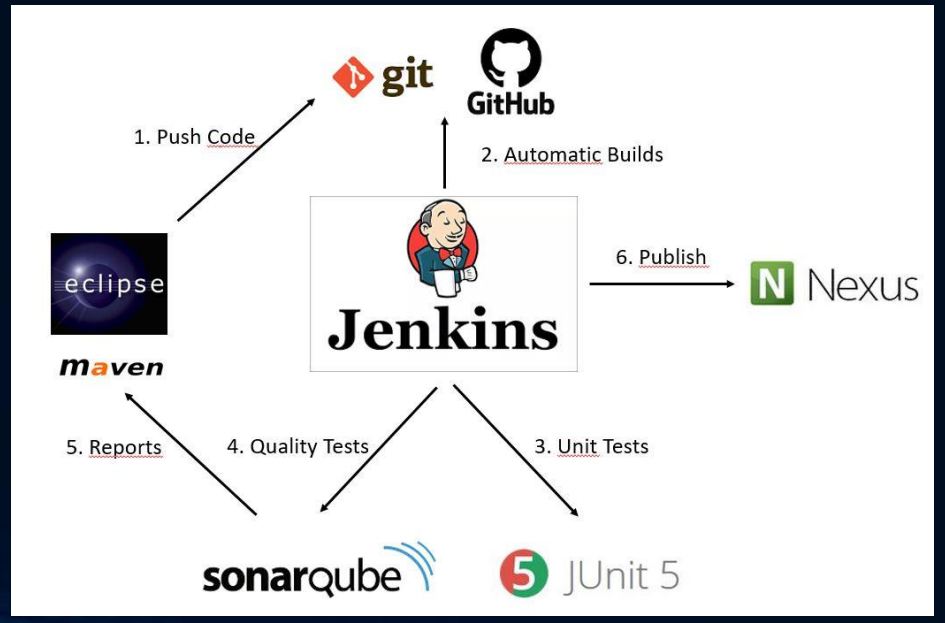
SCRUM

Es el marco Agile más utilizado, Es un ciclo de vida iterativo e incremental, con iteraciones cortas o sprints (semanas). Este tipo de ciclo de vida también se suele llamar adaptativo u orientado al cambio.

Fuente: https://www.itnove.com/es/agile/coaching-consultoria-scrum-barcelona

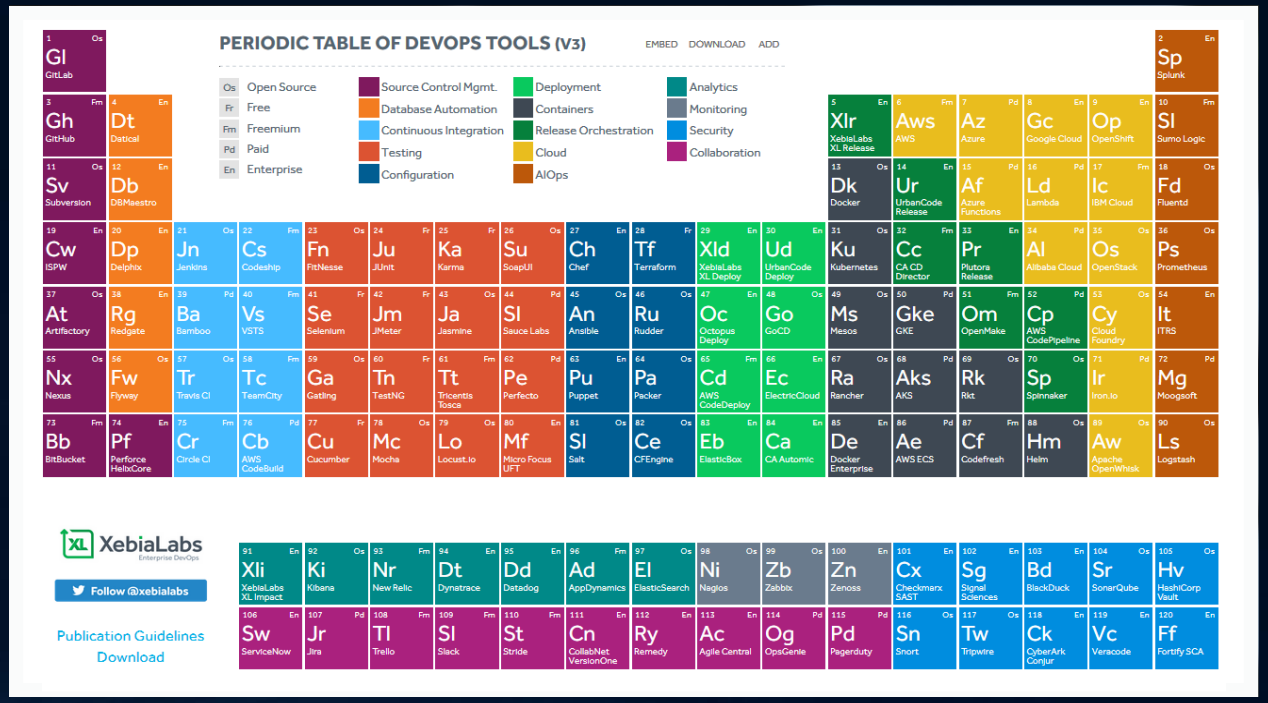
Ejemplo de entorno de Integración Continua (CI)

- Este ejemplo de Integración Continua funciona de manera que cada vez que un programador haga un cambio en el código del repositorio Git remoto, Jenkins lo detecta, compila la aplicación con Maven, llama a Junit para pasar los test unitarios y luego a SonarQube para ejecutar los test de calidad del código y, si todo ha ido bien, publica el resultado del proyecto en un repositorio de artefactos como Nexus.



Estado del arte de las herramientas DevOps (I)

En la sección 3.5 de la memoria se analiza el estado del arte de las herramientas más importantes utilizadas actualmente en cada una de las categorías involucradas en los procesos DevOps, centrándose en las herramientas Open Source.



Fuente: https://xebialabs.com/periodic-table-of-devops-tools/

Estado del arte de las herramientas DevOps (II)

Los tipos de herramientas Open Source analizados en la sección 3.5 de la memoria son las siguientes:

- Gestión de la configuración
- Integración Continua (CI)
- Testing
- Entrega y Despliegue Continuos
- Monitorización
- Alertas
- Seguridad
- Virtualización y Containerización
- Orquestación
- Comunicación y Colaboración

3. Introducción a DevSecOps

¿Qué es DevSecOps? (I)

El término DevSecOps se compone de tres palabras: Dev, que se refiere al equipo de desarrollo; Sec, que representa la seguridad; y Ops, que se refiere al equipo de operaciones.

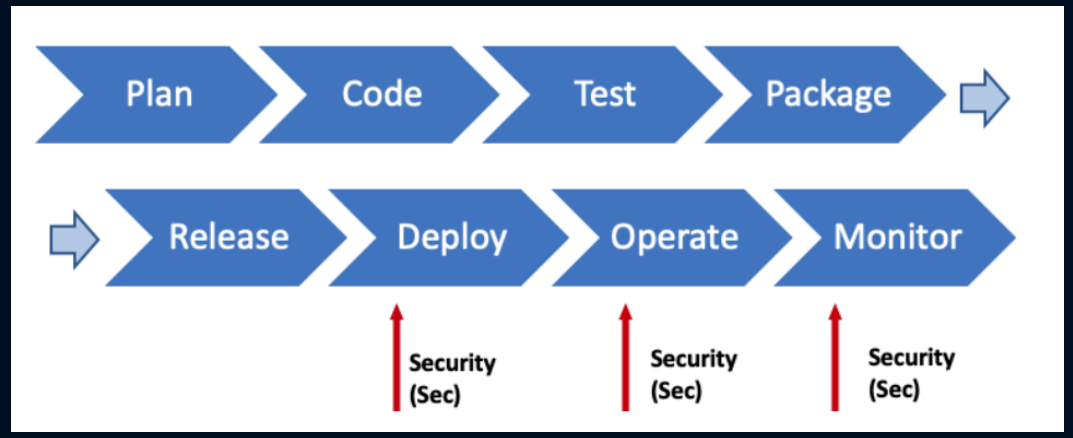
DevSecOps busca abordar los problemas de seguridad desde el inicio del ciclo de vida del desarrollo del software, integrando la seguridad en todas las fases del proceso de desarrollo de software y fomentando la colaboración entre equipos para mejorar la seguridad y la calidad del software entregado.

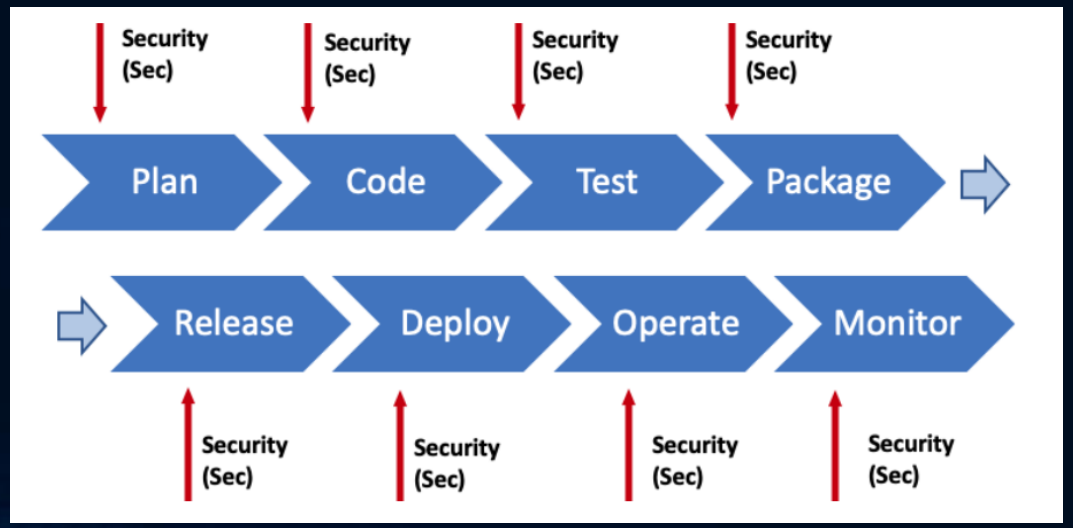
El objetivo principal de DevOps es abordar la seguridad desde el inicio del desarrollo.

¿Qué es DevSecOps? (II)

Hasta hace poco tiempo, no se involucraba a los equipos de Seguridad hasta la fase de Despliegue, demasiado tarde para solucionar los problemas de seguridad, que generan retrabajos y retrasos en las entregas.

DevSecOps fomenta que la seguridad sea consideraba en cada etapa.





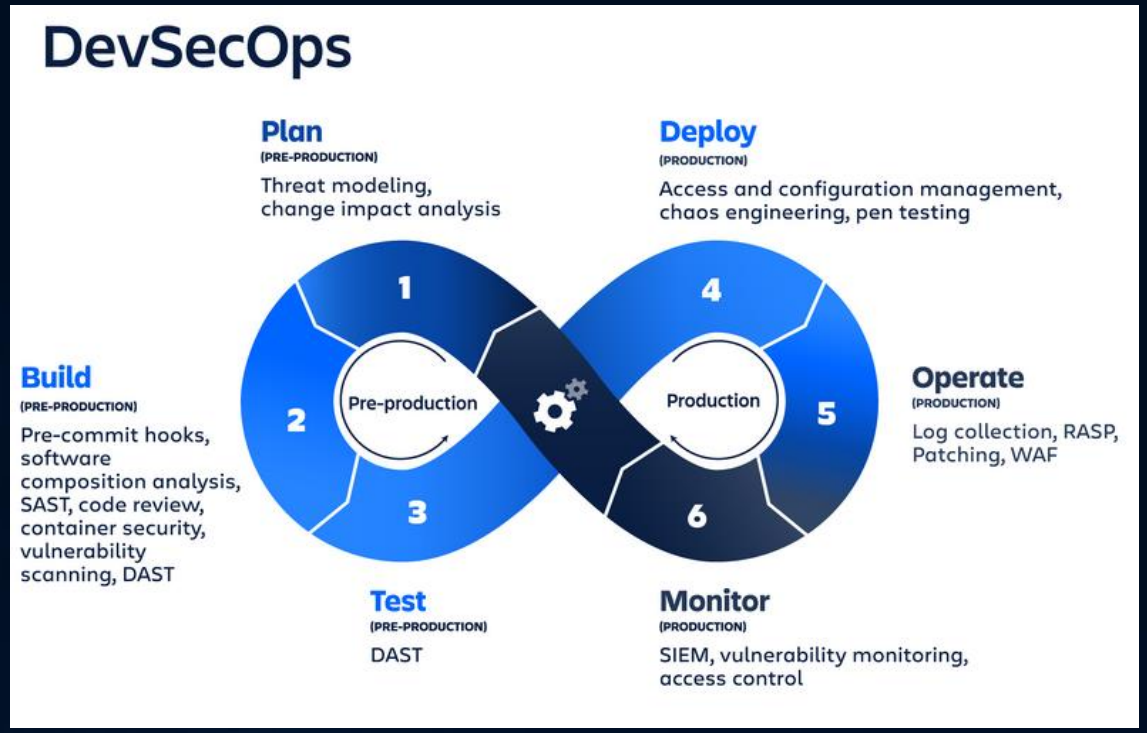
Fuente: https://www.elladodelmal.com/2019/02/secdevops-una-explicacion-en-cinco.html

DevSecOps vs SecDevOps

- DevSecOps: en lugar de aplicar la seguridad al producto final, tiene como objetivo tener la seguridad integrada en el producto aplicando el concepto de seguridad por diseño, lo que significa que la seguridad se toma en consideración desde el principio y durante todo el ciclo de vida del software que estemos desarrollando
- SecDevOps (Security DevOps): su objetivo principal es asegurar que los aspectos de seguridad se aborden desde las primeras etapas del desarrollo, en lugar de ser considerados como un paso posterior, involucrando a los equipos de seguridad de manera proactiva. Esto significa que la velocidad de lanzamiento ya no es una prioridad, lo que puede tener un efecto negativo en el proceso de desarrollo, debido a que se centra demasiado en la seguridad y no en que el producto sea entregado.

Ciclo de vida de DevSecOps (SSDLC)

En cada una de las etapas del SSDLC se debe de tener en cuenta la seguridad antes de pasar a la siguiente etapa.



Fuente: https://www.atlassian.com/es/devops/devops-tools/devsecops-tools

Modelado de amenazas

- El Modelado de amenazas (Threat Modeling) se lleva a cabo en las primeras etapas del ciclo de vida del desarrollo de software, como parte del diseño o planificación.
- El objetivo es identificar posibles vulnerabilidades o brechas de seguridad en el sistema y evaluar su impacto potencial.
- El Modelado de amenazas ayuda a los desarrolladores y a los equipos de seguridad a comprender cómo un atacante podría aprovechar una vulnerabilidad para acceder, dañar o robar información del sistema.
- Metodologías: las más conocidas son STRIDE, DREAD, SEI, OCTAVE y PASTA.

Análisis estático de la seguridad del código (SAST)

- Es una técnica que consiste en examinar el código fuente en busca de patrones o estructuras que puedan indicar una posible vulnerabilidad, defecto o mala práctica.
- Dentro del análisis estático del código también se pueden encontrar herramientas de cobertura de código testeado, test-coverage tools, que se utilizan para medir la cobertura de las pruebas en el código y garantizar que todas las funciones y líneas de código estén siendo evaluadas adecuadamente.

Análisis de secretos

- Actualmente, existe una proliferación no gestionada de secretos o credenciales (como contraseñas, claves de API, tokens, etc.) en diferentes partes del código, sistemas y herramientas utilizados en el proceso de desarrollo y despliegue de aplicaciones. Es lo que se conoce con el término “secret sprawl".
- Para evitar el "secret sprawl", se utilizan herramientas de gestión de secretos y prácticas de seguridad adecuadas, como el cifrado de secretos, la rotación regular de credenciales y la limitación de acceso a los secretos solo a las personas y sistemas autorizados.

Análisis de componentes del software (SCA)

- El análisis de componentes de software se realiza escaneando los componentes de software de terceros en un desarrollo software para identificar cualquier vulnerabilidad conocida o debilidad en la versión utilizada.
- Los resultados de este análisis se comparan con una base de datos de vulnerabilidades conocidas para determinar si existe algún riesgo de seguridad.
- Si se detecta una vulnerabilidad, se toman medidas para remediar el problema, como actualizar el componente a una versión más segura o buscar una alternativa más segura.

Análisis dinámico de la seguridad de las aplicaciones (DAST)

- Es una técnica que consiste en identificar vulnerabilidades de seguridad en una aplicación en tiempo de ejecución, mediante la realización de pruebas que simulan ataques externos a la aplicación.
- Este tipo de pruebas son conocidas como "black-box" ya que se parte de la premisa que se desconoce el funcionamiento interno de la aplicación o no se tiene control sobre él, es decir, no se puede depurar o interactuar con el código en funcionamiento.
- Normalmente, este tipo de pruebas se realizan de forma automatizada utilizando herramientas especializadas y también de forma manual por parte de pentesters o auditores de seguridad.

Análisis de la infraestructura como código (IAC)

- Es una técnica que analiza la seguridad de la IAC mediante el uso de herramientas que examinan el código utilizado para crear y configurar la infraestructura, para asegurar que la infraestructura utilizada para ejecutar las aplicaciones esté configurada de manera segura y para prevenir problemas de seguridad en la producción, como por ejemplo que no haya vulnerabilidades que puedan ser explotadas por atacantes.

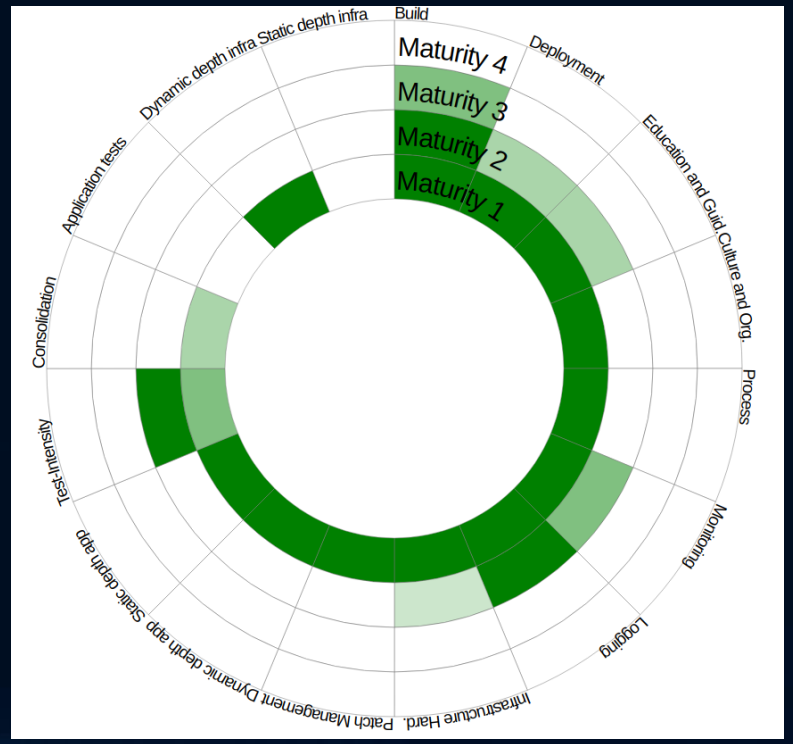
4. Aplicación práctica de implementación de DevSecOps

DevSecOps Maturity Model (DSOMM)(I)

- Es un marco de referencia que proporciona una guía estructurada para permitir a las organizaciones evaluar y mejorar gradualmente la incorporación de prácticas de seguridad en sus procesos de desarrollo y operaciones (DevOps).
- El objetivo del DevSecOps Maturity Model (DSOMM) es ayudar a las organizaciones a evaluar su nivel de madurez en cuanto a la implementación de prácticas de seguridad en su proceso de desarrollo y operaciones.

DevSecOps Maturity Model (DSOMM)(II)

- Proporciona una serie de niveles de madurez que van desde un enfoque inicial hasta un enfoque completamente integrado de seguridad en todas las actividades de DevOps.



Fuente: https://owasp.org/www-project-devsecops-maturity-model/

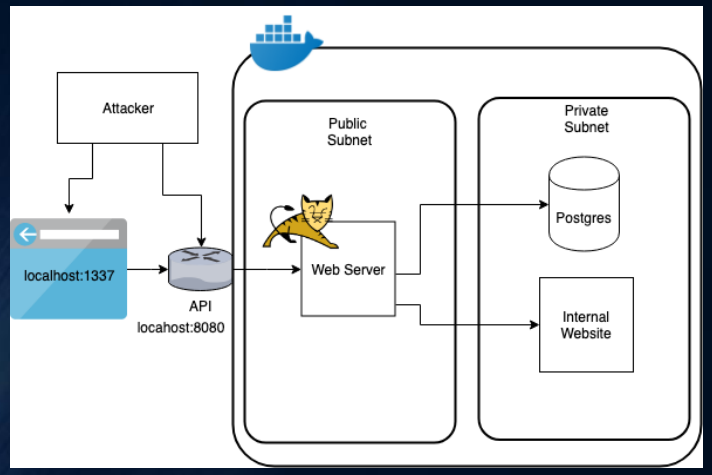
DevSecOps Maturity Model (DSOMM)(III)

- El cuarto objetivo es desarrollar un ejemplo de implantación de un proyecto de desarrollo de software cubriendo todo el ciclo de vida DevSecOps (SSDLC), empleando herramientas Open Source, de manera que cualquier empresa pueda utilizarlo, adaptándolo a sus necesidades, y alcance de esa manera un DevSecOps Maturity Model Level 1 (DSOMM Level1).
- Consolidar el DSOMM Level 1 exige la ejecución de herramientas de análisis estático (SAST), la detección de secretos y el análisis de componentes de software (SCA), sin ningún cambio en las herramientas o la configuración de las herramientas que se estés utilizando en el desarrollo de software.
- También se deben ejecutar pruebas con herramientas DAST, pero con su configuración de referencia.
- La ejecución de estas pruebas de seguridad se puede realizar una vez a la semana o al mes y los informes de las distintas herramientas pueden estar separados.
- Para conseguir este objetivo todas estas pruebas de seguridad se van a incluir en un pipeline completo de Jenkins de manera que cualquier empresa que desarrolle software, pueda adoptarlo en su propio pipeline de Jenkins para alcanzar de una manera sencilla un DSOMM Level 1.

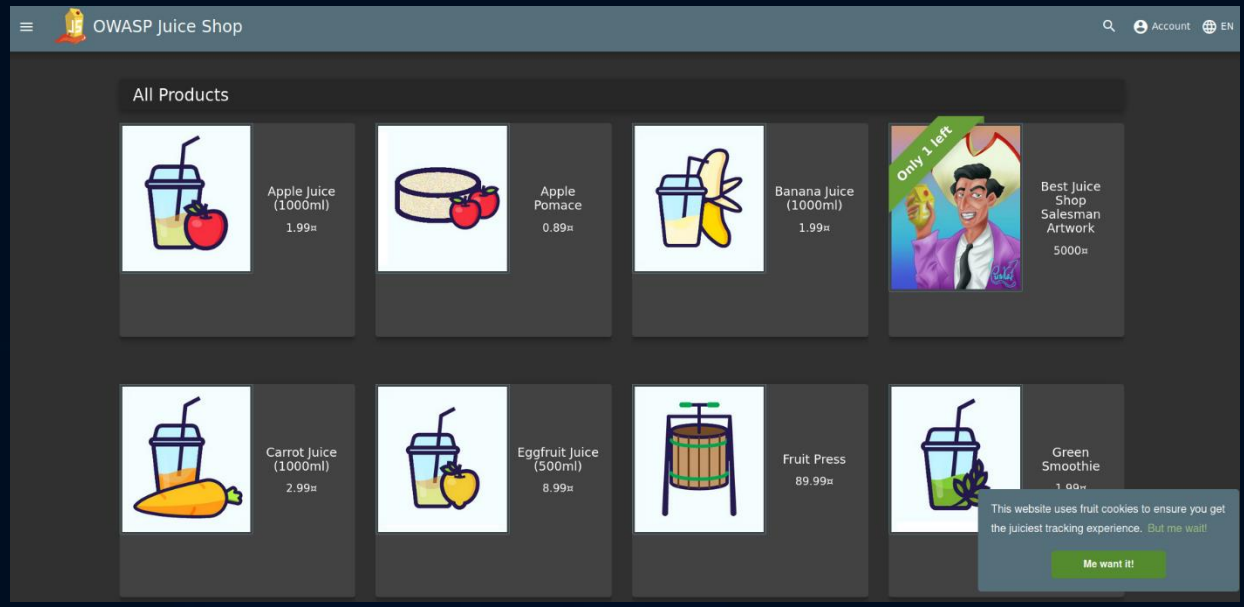
Aplicaciones a analizar en este TFG

Se van a utilizar dos aplicaciones para la realización de las pruebas :

- SAST => Vulnado
- DAST => OWASP Juice-Shop



Fuente: https://github.com/ScaleSec/vulnado



OWASP Juice Shop

Manual de instalación de las herramientas de seguridad

Para la realización del TFG se ha creado una máquina virtual (VM) en Oracle VirtualBox corriendo LINUX Ubuntu 22.04.2 LTS.

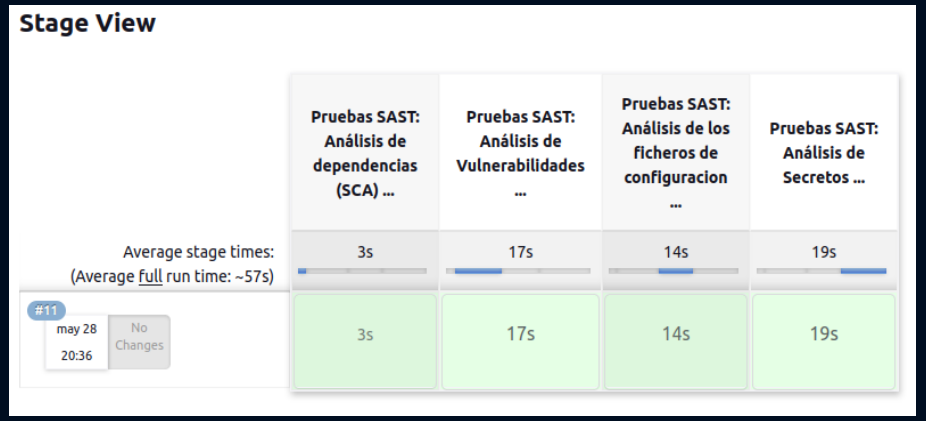
Se ha creado un manual de instalación para LINUX de todas las herramientas empleadas en los análisis de seguridad:

- Java
- Docker
- Docker-Compose
- Jenkins
- SonarQube
- Retire.JS
- OWASP Dependency-Check
- NodeJSScan
- Nuclei
- Wapiti
- ZAP Proxy
- Trivy

Pipelines de pruebas de seguridad en Jenkins

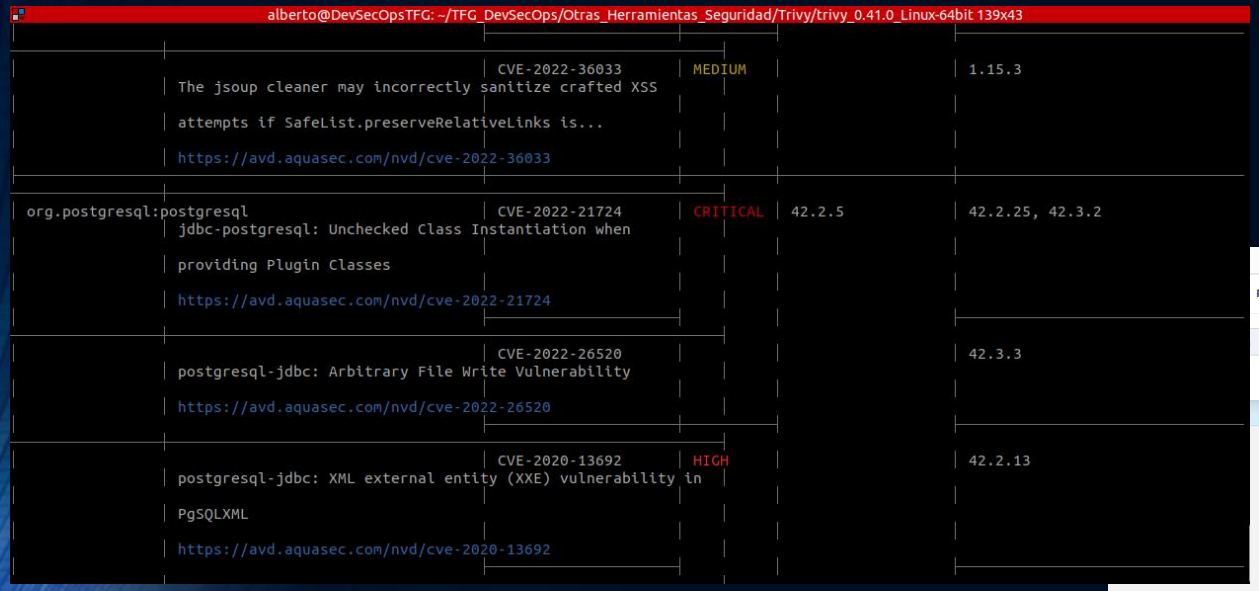
Se ha creado un pipeline Jenkins completo para cada aplicación/tipo de prueba: uno para Vulnado con pruebas SAST y otro para Juice-Shop para pruebas DAST.

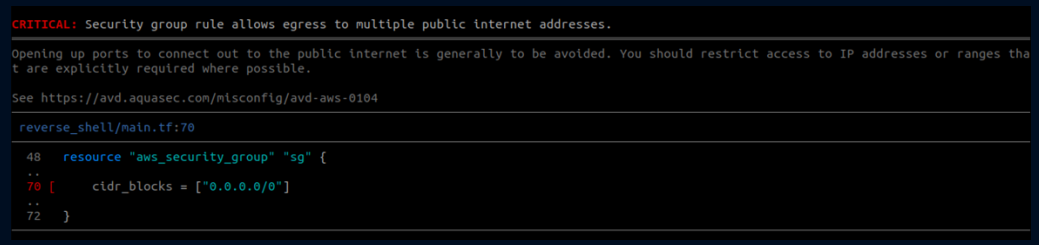
```
pipeline {
    // -----------------------------------------------------
    // -- Declaración de los trabajos se pueden ejecutar con cualquier nodo
    // -----------------------------------------------------
    agent any

    // -----------------------------------------------------
    // -- Declaración de las variables de entorno
    // -----------------------------------------------------
    environment {
    PATH_APP_VULNADO = '/home/alberto/TFG_DevSecOps/Aplicaciones_Analizadas/vulnado'

    PATH_DEPENDENCY_CHECK = '/home/alberto/TFG_DevSecOps/SAST/dependency-check/bin'
    OUTPUTS_DEPENDENCY_CHECK = '/tmp/Security_Reports/'

    OUTPUTS_RETIRE = '/tmp/Security_Reports/SecurityReport_Retire'

    PATH_TRIVY = '/home/alberto/TFG_DevSecOps/Otras_Herramientas_Seguridad/Trivy/trivy_0.41.0_Linux-64bit'
    OUTPUTS_TRIVY_CONFIG = '/tmp/Security_Reports/SecurityReport_Configuration_Trivy.html'
    OUTPUTS_TRIVY_VULN = '/tmp/Security_Reports/SecurityReport_Vulnerabilities_Trivy.html'
    OUTPUTS_TRIVY_SECRETS = '/tmp/Security_Reports/SecurityReport_Secrets_Trivy.html'

    PATH_SONARSCANNER = '/home/alberto/TFG_DevSecOps/SAST/SonarQube/sonar-scanner/bin'
    SONAR_PROJECT_KEY = credentials('SONAR_PROJECT_KEY')
    SONAR_TOKEN = credentials('SONAR_TOKEN')
    SONAR_URL = 'http://localhost:9000'
    }

    stages {
        // -----------------------------------------------------
        // -- Borrado de los reports de seguridad anteriores
        // -----------------------------------------------------
        stage ('Pruebas SAST: Borrado del contenido del directorio ... '){
                steps {
                  sh('rm -f /tmp/Security_Reports/*.*')
                }
        }


        // -----------------------------------------------------
        // -- Ejecución de Pruebas SAST: Análisis de dependencias (SCA) y Vulnerabilidades con Dependency-check
        // -----------------------------------------------------
        stage ('Pruebas SAST: Análisis de Vulnerabilidades con Dependency-check') {
            steps {
                sh('PATH_DEPENDENCY_CHECK/dependency-check.sh --scan $PATH_APP_VULNADO -o $OUTPUTS_DEPENDENCY_CHECK')
            }
        }

```

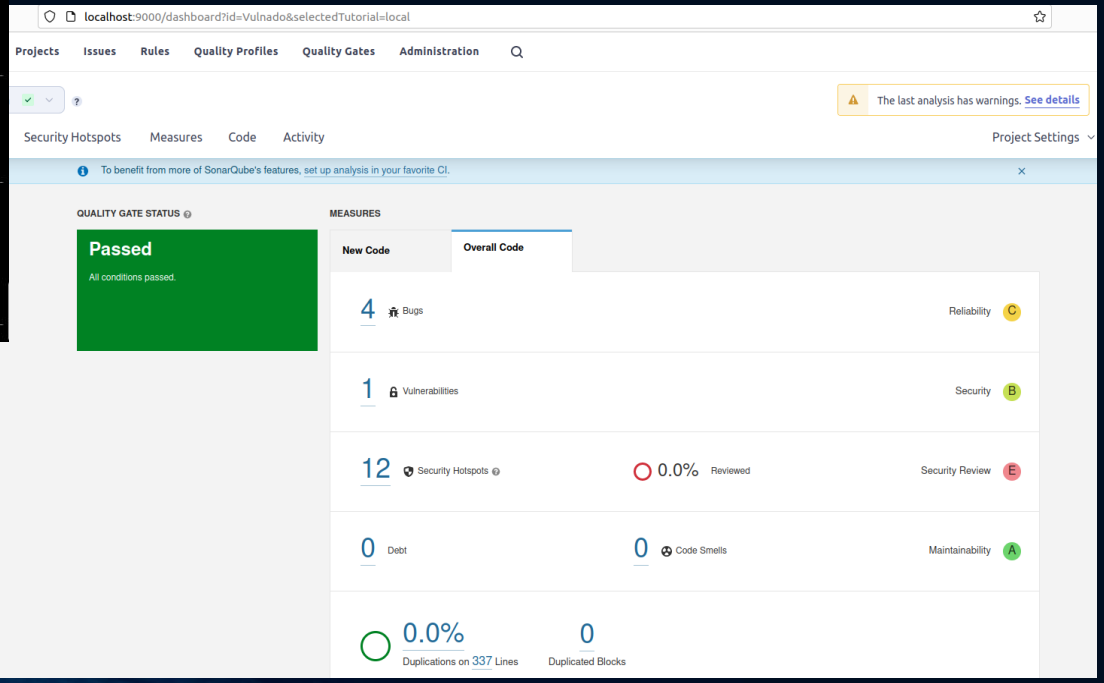


Análisis de los resultados de las pruebas de seguridad

Para cada prueba de seguridad se ha mostrado un ejemplo de los informes obtenidos.







5. Conclusiones

Conclusiones

- DevSecOps surge como una evolución natural de la metodología DevOps, incorporando un enfoque proactivo en la seguridad durante todo el ciclo de vida del desarrollo de software.
- Al igual que DevOps, se basa en la colaboración y comunicación entre los equipos de desarrollo y operaciones, pero con el añadido fundamental de integrar la seguridad desde el inicio.
- La adopción creciente de DevSecOps en las empresas de desarrollo de software se debe a los beneficios que aporta en términos de reducción de riesgos y brechas de seguridad, así como en la mejora de la calidad del software entregado.

Conclusiones del trabajo realizado en el TFG

- Todos los objetivos planteados al inicio del TFG se han cumplido.
- La planificación inicial se ha cumplido, sin desviaciones relevantes a destacar.
- Para todos los que lean este TFG espero que su lectura sea didáctica y les ayude a adoptar los principios de DevOps y DevSecOps, y que la parte práctica desarrollada les sirva como punto de partida para sus propios procesos de desarrollo internos.

Líneas futuras de desarrollo

- Incluir otro anexo con el proceso de instalación de las mismas herramientas de seguridad en Windows 11 para que lo puedan utilizar más empresas.
- Incluir en el pipeline un proceso completo de DevOps, que incluya el Deployment, para así tener un ciclo completo de desarrollo de software seguro.
- A nivel personal, el trabajo realizado durante este TFG ha sido muy satisfactorio, ya que me ha permitido ampliar mis conocimientos sobre DevOps, conocer lo que es DevSecOps y trabajar con herramientas con las que nunca antes había trabajado. Espero poder aplicar todo lo que he aprendido de DevSecOps en los proyectos en los que actualmente trabajo.

Agradecimientos

- Dedico este trabajo final de carrera a mis padres, las mejores personas que conozco, que nunca han tenido nada suyo, ya que lo poco que han tenido nos lo han dado a mi hermana y a mí. Ellos son el espejo donde me gustaría reflejarme cada día.
- También a mis suegros que son trabajadores incansables y siempre nos están ayudando en todo lo que pueden.
- Se lo dedico también a mi mujer Silvia, mi punto de apoyo en esta vida, sin su cariño y amor no podría vivir, y a mis hijos Alberto y Marcos, que son los dos soles que iluminan mi vida.
- A todos ellos les pido perdón por las ausencias, por todo el tiempo que he tenido que dedicar al proyecto y no a ellos.
- Quisiera darle las gracias a mi consultor, Joaquín López Sánchez-Montañés, por todo lo que me ha ayudado y todo lo que me ha apoyado durante la realización de este TFG.

Gracias por su atención

¿Preguntas?