

Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Proyecto Final

Seguridad K8s

Autor del proyecto: Andrea Álvaro Martín

Director del proyecto: María Francisca Pérez Pérez

Zaragoza, 26 de junio de 2023



Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma

Fecha **26 junio 2023**

Dedicatoria y Agradecimientos

"A todos mis profesores y la Universidad San Jorge por brindarme la oportunidad de sumergirme en el mundo de la tecnología y por haber fomentado, el desarrollo de mi curiosidad y por formarme como profesional y persona.

A María Francisca, directora de este proyecto, por acompañarme, guiarme, aconsejarme y animarme durante en este camino.

A NTTDATA, por brindarme un entorno de trabajo en el que he podido aprender y crecer.

A mi familia, sobre todo a mis padres, mi hermana y mis abuelos, gracias por vuestro esfuerzo y apoyo en mis decisiones. Nada de esto habría sido posible sin vosotros.

A mi pareja y amigos, que han estado a mi lado en todo momento, apoyándome y animándome a seguir adelante. No puedo estar más agradecida.

El camino llega a su fin y al fin resulta mi duda y un paso más cerca de formar parte del maravilloso mundo de la informática. Solo puedo decir ¡Gracias!"

Tabla de contenido

Resumen	1
Abstract	1
1. Introducción	3
2. Estado del Arte.....	5
2.1 De sistemas monolíticos a microservicios	5
2.2 Contenedores y Orquestación de contenedores	7
2.2.1 <i>Herramientas existentes.....</i>	<i>8</i>
2.2.2 <i>DevOps y Soluciones de seguridad en el mercado actual.....</i>	<i>9</i>
3. Objetivos.....	13
4. Metodología	15
4.1 Extreme Programming (XP).....	15
4.2 Aplicación de la tecnología	16
4.3 Seguimiento del desarrollo.....	16
4.3.1 <i>Trello.....</i>	<i>16</i>
4.3.2 <i>Diagrama de trabajo.....</i>	<i>17</i>
5. Análisis & Implementación.....	19
5.1 Iteración 1: Análisis	19
5.1.1 <i>Tareas US 0</i>	<i>19</i>
5.1.2 <i>Análisis del problema</i>	<i>19</i>
5.1.3 <i>Análisis de la guía de instalación.....</i>	<i>20</i>
5.2 Iteración 2 – Investigación & Análisis de Kubernetes	20
5.2.1 <i>Tareas US 1 y US 2.....</i>	<i>20</i>
5.2.2 <i>Desarrollo</i>	<i>20</i>
5.2.3 <i>Resultado.....</i>	<i>23</i>
5.3 Iteración 3 – Análisis de las características de seguridad en Kubernetes	24
5.3.1 <i>Tareas US 3</i>	<i>24</i>
5.3.2 <i>Desarrollo</i>	<i>24</i>
5.3.3 <i>Resultados</i>	<i>31</i>
5.4 Iteración 4. Investigación de herramientas de seguridad para Kubernetes. 33	
5.4.1 <i>Tareas US 4 y US 5.....</i>	<i>33</i>
5.4.2 <i>Desarrollo</i>	<i>33</i>
5.4.3 <i>Resultados</i>	<i>33</i>
5.5 Iteración 5 – Análisis de las herramientas seleccionadas – Cluster layer	34
5.5.1 <i>Tareas US 6 y US 7.....</i>	<i>34</i>
5.5.2 <i>Desarrollo</i>	<i>34</i>
5.5.3 <i>Resultados</i>	<i>42</i>
5.6 Iteración 6. Análisis de las herramientas en tiempo de ejecución	47
5.6.1 <i>Tareas US 8 y US 9.....</i>	<i>47</i>
5.6.2 <i>Desarrollo</i>	<i>47</i>
5.6.3 <i>Resultados</i>	<i>52</i>
5.7 Iteración 7 – Análisis Falco.....	53
5.7.1 <i>Tareas US 10 y US 14.....</i>	<i>53</i>
5.7.2 <i>Desarrollo</i>	<i>53</i>
5.7.1 <i>Resultados</i>	<i>55</i>
5.8 Iteración 8 – Análisis StackRox	55

5.8.1	Tareas US 11 y US 14	55
5.8.2	Desarrollo	56
5.8.3	Resultados	58
5.9	Iteración 9 – Análisis Sysdig Secure	58
5.9.1	Tareas US 12 y US 14	58
5.9.2	Desarrollo	58
5.9.3	Resultados	59
5.10	Iteración 10 – Análisis de las herramientas: Falco, StackRox, Sysdig.....	60
5.10.1	Tareas US 13.....	60
5.10.2	Desarrollo	60
5.10.3	Resultados	60
5.11	Iteración 11 – Redacción guía de instalación.....	63
5.11.1	Tareas US 14.....	63
5.11.2	Desarrollo	63
5.11.3	Resultado.....	64
5.12	Iteración 12 – Preparación para de demostración.....	64
5.12.1	Tareas US 15.....	64
5.12.2	Desarrollo	64
5.12.3	Resultados	64
6.	Estudio económico	65
6.1.1	Costes materiales.....	65
6.1.2	Costes humanos.....	65
6.1.3	Costes de infraestructura	66
6.1.4	Costes totales.....	66
7.	Resultados	67
8.	Conclusiones	71
8.1	Propuesta de mejora	71
9.	Bibliografía.....	73
Anexo 1:	Propuesta	76
Anexo 2:	Historias de Usuario.....	77
Anexo 3:	Iteraciones	81
Anexo 4:	Código YAML.....	84
Anexo 5:	Guía de instalación	91

Tabla de ilustraciones

ILUSTRACIÓN 1. CIBER AMENAZAS EN TIEMPO REAL SEGÚN KASPERSKY	3
ILUSTRACIÓN 2. COMPARATIVA DE SISTEMA MONOLÍTICOS Y MICROSISTEMAS	6
ILUSTRACIÓN 3. COMPARATIVA DE MÁQUINAS VIRTUALES Y CONTENEDORES	8
ILUSTRACIÓN 4. CICLO DE DESARROLLO DE SOFTWARE EN DEVOPS.....	10
ILUSTRACIÓN 5. CLASIFICACIÓN DE HERRAMIENTAS DE SEGURIDAD EXISTENTES	11
ILUSTRACIÓN 6. TRELLO ´S DASHBOARD DEL PROYECTO.....	17
ILUSTRACIÓN 7. DIAGRAMA DE GANTT DEL PROYECTO.....	18
ILUSTRACIÓN 8. KUBERNETES DASHBOARD	22
ILUSTRACIÓN 9. ARQUITECTURA KUBERNETES.....	23
ILUSTRACIÓN 10. LAS 4C DE LA SEGURIDAD NATIVA EN LA NUBE.....	25
ILUSTRACIÓN 11. SELECCIÓN DE HERRAMIENTAS A ANALIZAR	34
ILUSTRACIÓN 12. EJECUCIÓN KUBE-BENCH	36
ILUSTRACIÓN 13. RESULTADOS DE LA HERRAMIENTA KUBE-BENCH.....	36
ILUSTRACIÓN 14. EJECUCIÓN KUBE-HUNTER.....	37
ILUSTRACIÓN 15. RESULTADOS DE LA HERRAMIENTA KUBE-HUNTER	37
ILUSTRACIÓN 16. EJECUCIÓN DE VAULT	38
ILUSTRACIÓN 17. DASHBOARD VAULT	39
ILUSTRACIÓN 18. EJEMPLO VAULT.....	39
ILUSTRACIÓN 19. EJECUCIÓN SEALED SECRETS	40
ILUSTRACIÓN 20. APLICAR POLÍTICA EN OPA	41
ILUSTRACIÓN 21. EJEMPLO APLICACIÓN DE POLÍTICA	42
ILUSTRACIÓN 22. RESULTADOS TRIVY	50
ILUSTRACIÓN 23. FORMATO ALERTA EN FALCO	54
ILUSTRACIÓN 24. DASHBOARD REDHAT STACKROX.....	57
ILUSTRACIÓN 25. PLANIFICACIÓN INICIAL Y FINAL	68
ILUSTRACIÓN 26. PLANIFICACIÓN TOTAL	68
ILUSTRACIÓN 27. CONFIG.YAML INSTALACIÓN KUBE-BENCH	84
ILUSTRACIÓN 28. SEALED-SECRET.YAML FILE.....	85
ILUSTRACIÓN 29. POLÍTICA DE TIPO "GENERATE": AÑADIR POLÍTICAS DE RED	86
ILUSTRACIÓN 30. POLÍTICA DE TIPO "MUTATE": AÑADIR ETIQUETAS	87
ILUSTRACIÓN 31. POLÍTICA DE TIPO "VALIDATE": PERMITE ANOTACIONES	88
ILUSTRACIÓN 32. POLÍTICA DE TIPO "VERIFY IMAGES": REQUIERE DE ESCANEADO DE VULNERABILIDADES.....	89
ILUSTRACIÓN 33. EJEMPLO DE POLÍTICA EN GATEKEEPER.....	90
ILUSTRACIÓN 34. GUÍA DE INSTALACIÓN PARA MACOS	91

Tabla de tablas

TABLA 1. COMPARATIVA DE MÉTODOS DE INSTALACIÓN EN KUBERNETES	24
TABLA 2. CARACTERÍSTICAS DE SEGURIDAD EN KUBERNETES.....	32
TABLA 3. COMPARATIVA KUBE-BENCH Y KUBE-HUNTER	43
TABLA 4. COMPARATIVA SEALED SECRETS Y VAULT.....	45
TABLA 5. COMPARATIVA KYVERNO Y OPEN POLICY AGENT	46
TABLA 6. COMPARATIVA OPA GATEKEEPER Y KYVERNO VENTAJAS Y DESVENTAJAS	47
TABLA 7. COMPARATIVA CLAIR Y TRIVY	52
TABLA 8. COMPARATIVA FALCO, STACKROX Y SYSDIG SECURE.....	61
TABLA 9. CASOS DE USO DE LAS HERRAMIENTAS FALCO, STACKROX Y SYSDIG SECURE	62
TABLA 10. CASOS DE USO DEPENDIENDO DEL PROYECTO: FALCO, STACKROX Y SYSDIG SECURE.....	62
TABLA 11. COSTES MATERIALES	65
TABLA 12. COSTES HUMANOS.....	65
TABLA 13. COSTES DE INFRAESTRUCTURA	66
TABLA 14. COSTES TOTALES	66

Resumen

En los últimos años, la adopción de arquitecturas basadas en microservicios ha aumentado significativamente, lo que ha creado una necesidad de herramientas de seguridad especializadas para proteger plataformas basadas en Kubernetes (K8s).

La seguridad en Kubernetes es crucial ya que una brecha de seguridad puede tener graves consecuencias como pérdida de datos o interrupción del servicio, lo que dañaría la reputación de la organización. Los entornos de Kubernetes son altamente dinámicos y escalables, lo que los hace vulnerables a las amenazas de seguridad. Es necesario contar con herramientas actualizadas y efectivas para detectar y mitigar los riesgos de seguridad de manera oportuna en la nube. Por lo tanto, el objetivo de este proyecto es analizar y comparar herramientas de seguridad para K8 como Falco, StackRox y Sysdig Secure.

El proyecto se divide en dos partes. En la primera sección, se realizará un estudio técnico de las herramientas de seguridad disponibles para Kubernetes, con un enfoque en herramientas en tiempo de ejecución como Falco, StackRox y Sysdig Secure para determinar cuál es el mejor en cada caso de uso. En la segunda sección, las herramientas elegidas para la demostración se desempaquetarán en un entorno Kubernetes, y se creará una guía de instalación para los usuarios.

Palabras clave: *Kubernetes, seguridad, microservicios, tiempo de ejecución, guía instalación de herramientas K8s.*

Abstract

In recent years, the adoption of microservices-based architectures has significantly increased, creating a need for specialized security tools to protect Kubernetes-based platforms (K8s).

Security in Kubernetes is crucial as a security breach can have severe consequences such as data loss or service disruption, which would damage the organization's reputation. Kubernetes environments are highly dynamic and scalable, making them vulnerable to security threats. It is necessary to have up-to-date and effective tools to timely detect and mitigate security risks in the cloud. Therefore, the objective of this project is to analyze and compare security tools for K8s such as Falco, StackRox, and Sysdig Secure.

The project is divided into two parts. In the first section, a technical study of the available security tools for Kubernetes will be conducted, with a focus on runtime tools like Falco, StackRox, and Sysdig Secure to determine the best tool for each use case. In the second section, the chosen tools for the demonstration will be unpacked in a Kubernetes environment, and an installation guide will be created for the users.

Keywords: *Kubernetes, security, microservices, runtime, K8s tool installation guide.*

1. Introducción

En la actualidad, la tecnología de contenedores se ha convertido en una herramienta fundamental para el despliegue de aplicaciones y servicios en entornos de nube. En este contexto, Kubernetes ha surgido como la plataforma líder en la orquestación de contenedores, siendo ampliamente adoptada en todo el mundo. Sin embargo, la seguridad en Kubernetes se ha convertido en un tema crítico y de gran preocupación en los últimos años. Según Kaspersky [1], España se encuentra entre los 12 países más atacados, lo que subraya aún más la importancia de abordar la seguridad en entornos de Kubernetes [1]. En la ilustración 1, se muestra una representación de los ataques en tiempo real dirigidos a dispositivos ubicados en España. La abundancia de estos ataques resalta la importancia de la seguridad en la sociedad actual.

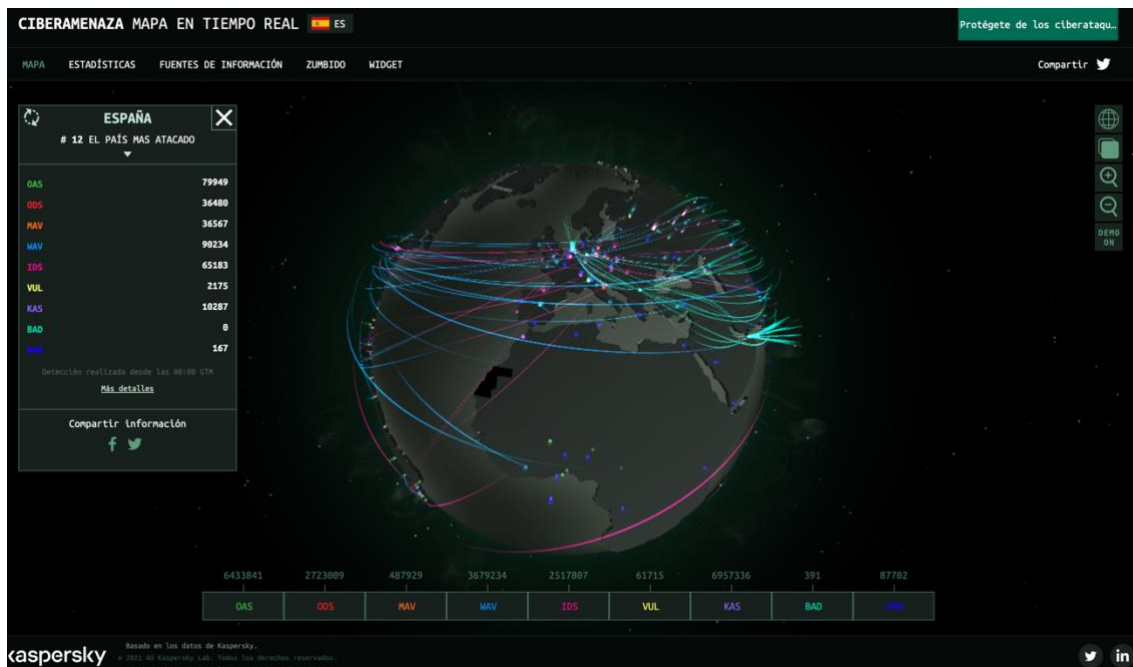


Ilustración 1. Ciber amenazas en tiempo real según Kaspersky

El problema principal radica en que, al igual que cualquier otra tecnología, Kubernetes presenta vulnerabilidades y debilidades que pueden ser explotadas por actores malintencionados. La naturaleza distribuida y escalable de Kubernetes agrega complejidad a la tarea de proteger y monitorear adecuadamente los entornos desplegados. Además, dado que Kubernetes se utiliza en entornos empresariales críticos, cualquier violación de seguridad puede tener consecuencias graves y potencialmente catastróficas.

Por tanto, es crucial desarrollar estrategias y herramientas de seguridad efectivas para proteger de manera adecuada las implementaciones de Kubernetes. Esto permitirá mitigar los riesgos de ataques, fugas de información e interrupciones de servicio, asegurando la integridad y disponibilidad de los sistemas.

Con el objetivo de abordar esta problemática, el proyecto propuesto se enfocará en realizar una investigación exhaustiva y una evaluación comparativa de las herramientas de seguridad existentes, tanto a nivel de clúster como a nivel de aplicación en entornos de Kubernetes. El objetivo es identificar las soluciones más apropiadas y efectivas para enfrentar los desafíos de seguridad específicos de Kubernetes.

El proyecto se llevará a cabo en colaboración con la empresa NTTDATA, líder en el sector de la tecnología y la consultoría. Esta colaboración permitirá acceder a recursos adicionales, conocimientos especializados y una visión empresarial que enriquecerá el desarrollo del proyecto y garantizará su aplicabilidad y relevancia en el mundo real.

El proyecto se dividirá en tres etapas diferenciadas. En la primera etapa, se realizará un análisis exhaustivo y una selección de las herramientas más relevantes y ampliamente utilizadas en la protección de Kubernetes. Se presentarán de manera concisa las herramientas seleccionadas, proporcionando una visión general del contexto de Kubernetes y las opciones disponibles para su protección.

En la segunda etapa, se profundizará en el análisis de las herramientas seleccionadas. Se elaborarán tablas comparativas detalladas, destacando las características principales de las herramientas clave, como Falco, StackRox y Sysdig Secure. Además, se realizará una comparativa exhaustiva y se presentarán casos de uso específicos para cada herramienta.

En la tercera y última etapa, se desarrollará una guía de instalación detallada y se presentarán las mejores prácticas para la configuración y el despliegue seguro de Kubernetes. Además, se preparará un caso práctico que permitirá obtener un entorno de Kubernetes protegido y seguro.

En conclusión, el proyecto abordará el problema de seguridad en entornos de Kubernetes mediante una investigación exhaustiva de herramientas, una evaluación comparativa y la implementación de las mejores prácticas de seguridad. La colaboración con NTTDATA y el enfoque práctico y detallado del proyecto asegurarán su relevancia y aplicabilidad en el ámbito empresarial, contribuyendo así a mejorar la seguridad de los datos en la nube en los entornos de Kubernetes.

2. Estado del Arte

Hoy en día son más numerosas las organizaciones y empresas que buscan integrar un diseño a gran escala y de alto rendimiento. Es por ello por lo que muchas de ellas se decantan por una arquitectura enfocada a microservicios.

2.1 De sistemas monolíticos a microservicios

En los últimos años, los sistemas monolíticos han sido la principal opción utilizada por las empresas, y muchas siguen utilizándolos. Estos sistemas se fundamentan en una base de funcionalidades centralizada, donde todas las funcionalidades y servicios de negocio están agrupados en una base de código única. Entre sus principales ventajas destacan su facilidad de desarrollo inicial, desarrollo centralizado que puede establecer un estándar, y su simplicidad a nivel de despliegue y ejecución.

Sin embargo, para realizar un cambio en este tipo de arquitectura, es necesario actualizar todo el código base, creando y desplegando una versión actualizada de la aplicación. Esto hace que las actualizaciones sean restrictivas y consuman más tiempo de desarrollo, pruebas y despliegue. Además, se dificulta la adopción de nuevas tecnologías, se requiere un despliegue completo tras cada actualización y los fallos pueden propagarse por todo el sistema.

Aunque la arquitectura monolítica todavía es común para muchas aplicaciones, no es la mejor opción para sistemas complejos. A medida que el software evoluciona y crece, el coste de mantener una arquitectura única se vuelve cada vez mayor, mientras que los beneficios [2] de adoptar una arquitectura más flexible para respaldar e impulsar el crecimiento empresarial son aún mayores. La arquitectura de microservicios surgió como una respuesta de los desarrolladores a la incapacidad de continuar escalando aplicaciones.

Por ello, las arquitecturas de microservicios resuelven estos problemas mencionados, permitiendo así una sencillez en el desarrollo, test y despliegue, gran escalabilidad horizontal, aislamiento de errores, total adaptabilidad a nuevas tecnologías [2].

En la ilustración 2, se presenta de manera esquemática una comparación entre la arquitectura de un sistema monolítico y un sistema de microservicios, brindando una visión general de las diferencias entre ambas estructuras.

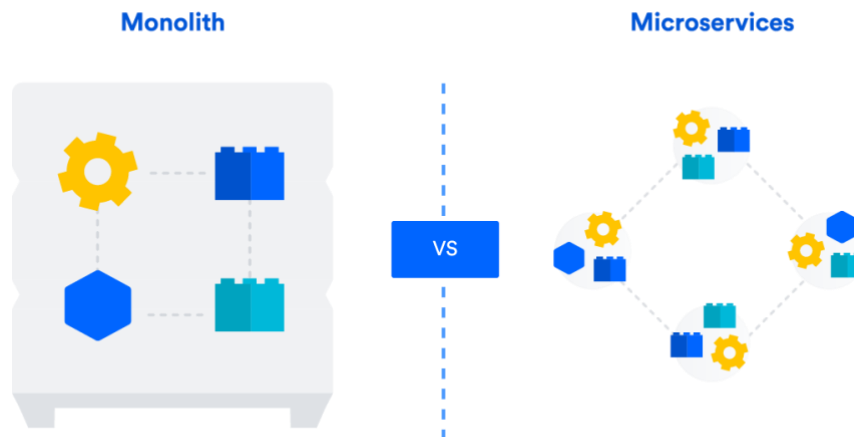


Ilustración 2. Comparativa de Sistema Monolíticos y Microsistemas

Las arquitecturas de microservicios resuelven los problemas mencionados, permitiendo un sencillo desarrollo, pruebas y despliegue, una gran escalabilidad horizontal, aislamiento de errores y total adaptabilidad a nuevas tecnologías.

No obstante, los microservicios también plantean desafíos que deben ser abordados. Uno de ellos es la necesidad de gestionar múltiples servicios, lo que puede aumentar la complejidad para los desarrolladores, operadores y *DevOps*. Además, es necesario coordinar los servicios entre sí para garantizar su correcto funcionamiento.

El diseño de aplicaciones basadas en microservicios puede ser complicado y requiere un diseño adecuado de los microservicios individuales. También puede haber complejidad en la consistencia de los datos y transacciones, ya que cada servicio tiene su propia base de datos. A medida que aumenta la necesidad de automatización, también aumenta la necesidad de una supervisión rigurosa.

Otro aspecto importante que considerar es el aumento de los riesgos de seguridad, ya que cada función se expone al exterior a través de una API, lo que resulta en un mayor número de posibles vectores de ataque. Esto ha llevado a la búsqueda de soluciones y análisis de amenazas en este ámbito.

Empresas destacadas como Netflix [2], Spotify [3], Adidas [4] o Amazon [5] utilizan arquitecturas basadas en microservicios en muchos de sus servicios y aprovechan Docker para acelerar el desarrollo y la implementación de aplicaciones. Esta arquitectura se utiliza para implementar y

ejecutar entornos en la nube, que ofrecen numerosas ventajas y oportunidades, pero también conllevan riesgos.

A medida que aumenta la complejidad de la infraestructura utilizada en la nube, también aumentan los riesgos, lo que significa que los ciberdelincuentes están aprovechando estas oportunidades para llevar a cabo ataques. Por lo tanto, es fundamental abordar la seguridad en Kubernetes, la plataforma de orquestación de contenedores de código abierto más utilizada para administrar aplicaciones basadas en microservicios.

2.2 Contenedores y Orquestación de contenedores

La tecnología de despliegue de aplicaciones basada en contenedores ha sido uno de los mayores avances en la industria del software en los últimos años. Debido a la necesidad de acelerar la actualización continua de aplicaciones y la irrupción de la metodología *DevOps*, estos despliegues se han vuelto cada vez más comunes. Muchas organizaciones y empresas se han inclinado por el uso de contenedores debido a su facilidad para el despliegue de nuevos servicios, la escalabilidad y la capacidad de adaptarse a cualquier entorno.

Los contenedores son una forma de virtualización del sistema operativo [6] que permite utilizar un único contenedor para ejecutar todo, desde microservicios o procesos de software hasta grandes aplicaciones. Contienen todos los archivos ejecutables, códigos binarios, bibliotecas y archivos de configuración necesarios, pero a diferencia de los métodos de virtualización de computadoras o servidores, no contienen imágenes del sistema operativo, lo que los hace más ligeros y reduce considerablemente el costo. En despliegues de aplicaciones más grandes, se pueden implementar varios contenedores como uno o más clústeres de contenedores.

Los contenedores ofrecen un mecanismo de empaquetado lógico que permite abstraer las aplicaciones del entorno en el que se ejecutan, lo que facilita y uniformiza el despliegue de aplicaciones basadas en contenedores. Si se compara con las máquinas virtuales, los contenedores permiten empaquetar las aplicaciones con bibliotecas y otros programas, proporcionando entornos aislados para ejecutar servicios software y simplificando aún más el entorno.

En resumen, los contenedores ofrecen una solución mucho más ligera que permite a los desarrolladores y equipos de operaciones de IT trabajar con mayor facilidad y disfrutar de ciertos beneficios. En la ilustración 3 [6], se muestra una comparativa estructural entre las máquinas virtuales y los contenedores. Es evidente que los contenedores son más sencillos, ya que cada uno está compuesto por una aplicación y los archivos binarios y librerías necesarias.

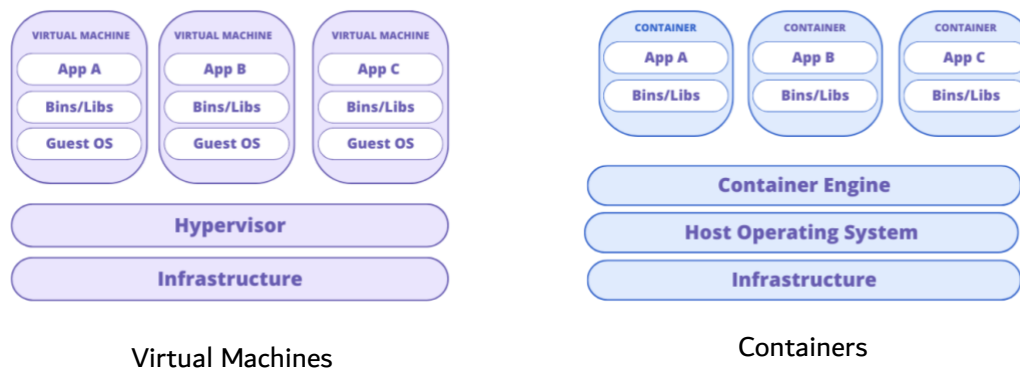


Ilustración 3. Comparativa de Máquinas virtuales y Contenedores

La integración de los contenedores y orquestadores de contenedores, así como la monitorización del sistema, permite que el despliegue de los servicios en un entorno de producción se haga de manera fiable y robusta.

Los contenedores aportan las características y herramientas concretas que se necesitan en la actualidad [6], donde la portabilidad, escalabilidad, alta disponibilidad y los microservicios en aplicaciones distribuidas son cada vez más utilizados. Cada vez se desarrollan menos aplicaciones monolíticas y más basadas en módulos y microservicios. Esto permite un desarrollo más ágil, más rápido y simultáneamente portátil.

2.2.1 Herramientas existentes

Docker

Docker [7] es la plataforma de open source más utilizada para la creación de contenedores, y permite a los usuarios empaquetar, distribuir y administrar aplicaciones dentro de contenedores. Con Docker, se puede implementar y escalar rápidamente sus aplicaciones a cualquier entorno con la confianza de que su código se ejecutará en cualquier lugar. Con Docker, los desarrolladores y equipos de operaciones de IT pueden implementar aplicaciones y servicios de forma rápida y eficiente, sin tener que preocuparse por las complejidades de la infraestructura subyacente.

Kubernetes

Kubernetes [8] es una plataforma de orquestación de contenedores de código abierto que permite la gestión automatizada de aplicaciones en contenedores en un entorno de nube o local. Fue desarrollado por Google y se convirtió en un proyecto de la *Cloud Native Computing Foundation*. Kubernetes proporciona un conjunto de características para desplegar, escalar y gestionar aplicaciones contenerizadas de manera eficiente y segura. Algunas de sus características principales incluyen el escalado automático de aplicaciones, la gestión de la configuración, la

tolerancia a fallos, el balanceo de carga y la gestión de recursos. También admite múltiples proveedores de nube y sistemas operativos, lo que lo hace altamente portátil y adaptable a diferentes entornos.

Según un informe de *Cloud Native Computing Foundation* publicado en 2020 [9], Kubernetes se ha convertido en la plataforma de orquestación de contenedores dominante en la nube, con el 83% de las empresas encuestadas. Además, el informe encuentra que el 78 % de las organizaciones que utilizan Kubernetes lo utilizan en un entorno de producción.

Otro estudio de *Datadog* publicado en 2021 [10] encuentra que el uso de Kubernetes ha aumentado significativamente en los últimos años. Según su informe, el uso de Kubernetes aumentó un 37% en 2020 en comparación con 2019 y su adopción sigue creciendo.

Además, según una encuesta de Red Hat de 2022 [11], el 94 % de los encuestados emplea Kubernetes en producción y el 89 % planea expandir su uso de Kubernetes en el futuro.

Además, según una encuesta de Enlyft [12], 76,020 compañías usan Kubernetes. Las compañías que más emplean Kubernetes se localizan en Estados Unidos y en los departamentos de servicios de industria y de información de la tecnología.

Por lo tanto, estos datos indican que Kubernetes se adopta y usa ampliamente en todo el mundo y que su adopción y uso continúan creciendo en la actualidad.

2.2.2 DevOps y Soluciones de seguridad en el mercado actual

Las empresas tienen la responsabilidad de mantener el sistema operativo actualizado e instalar los parches de seguridad que sean necesarios en todo momento. Al igual que ocurre en los servidores que son propiedad de una empresa, es necesario mantener políticas de seguridad tradicionales como el control de usuarios, la correcta configuración de servicios, o la revisión del software para comprobar que no tiene vulnerabilidades, entre otros aspectos.

Para intentar solventar los problemas de seguridad en los microservicios, han surgido diversas herramientas para automatizar y gestionar la seguridad de dichos entornos de una manera más rápida y eficaz.

Las vulnerabilidades de software están presentes desde los inicios de la informática, y en los últimos tiempos la preocupación por las mismas ha ido en aumento en todos los ámbitos debido a los daños materiales, económicos e incluso a la reputación que pueden ocasionar. Los

ciberdelincuentes y profesionales de la ciberseguridad estudian las vulnerabilidades con el fin de explotarlas en el caso de los ciberdelincuentes, y darles una solución, mitigar los daños o prevenir una posible incidencia en el caso de los profesionales.

En un entorno *DevOps* [13] basado en contenedores, las organizaciones deben abordar tres problemas de seguridad clave. En primer lugar, detectar vulnerabilidades en las aplicaciones tanto en el código fuente de la aplicación como en dependencias externas; por ejemplo, componentes open source. En segundo lugar, flexibilidad, es decir, que no requieran perímetros externos o configuración de redes. Finalmente, deben garantizar la seguridad durante todo el ciclo de vida del desarrollo de software.

En la ilustración 4 [15], se muestra de forma visual el ciclo de desarrollo de software aplicado en un entorno *DevOps* y las diferentes etapas que lo componen. Este ciclo se basa en una serie de etapas clave que se repiten de manera iterativa para lograr un flujo continuo y confiable en la entrega de software.

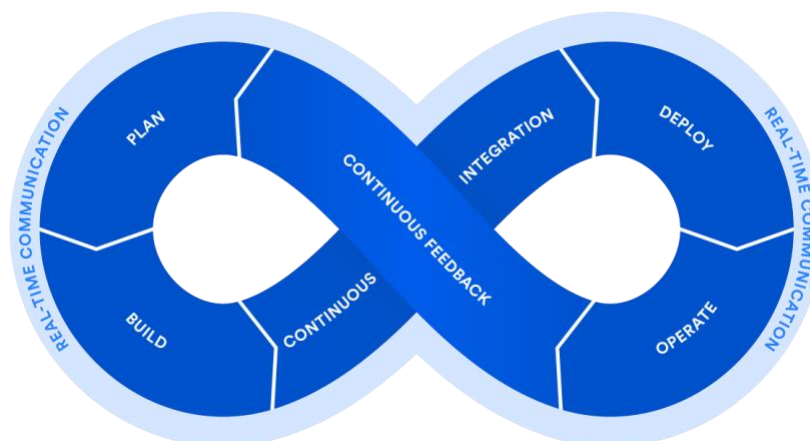


Ilustración 4. Ciclo de desarrollo de Software en *DevOps*

Es importante destacar que el ciclo de desarrollo de software en un entorno *DevOps* es cíclico y continuo, lo que significa que una vez que se completa una iteración, se vuelve al inicio del ciclo para comenzar una nueva planificación y desarrollo, con el objetivo de seguir mejorando el software y adaptándolo a las necesidades cambiantes de los usuarios.

Para garantizar la seguridad durante todo el desarrollo de software han surgido los perfiles *DevSecOps* [14], que se centran en añadir seguridad al ciclo de vida de software tanto en el despliegue como monitorización de los sistemas.

La seguridad de los contenedores es importante porque la imagen del contenedor contiene todos los componentes que eventualmente ejecutarán la aplicación. Si una imagen de contenedor

contiene vulnerabilidades, aumenta el riesgo y la gravedad potencial de los problemas de seguridad durante la producción y puede ser catastrófico para una empresa u organización. Actualmente en el mercado existen diversas herramientas y soluciones que proporcionan a las empresas llevar a cabo un ciclo de vida de software seguro.

En la ilustración 5, se muestra una clasificación de las herramientas de seguridad existentes, organizadas en categorías.

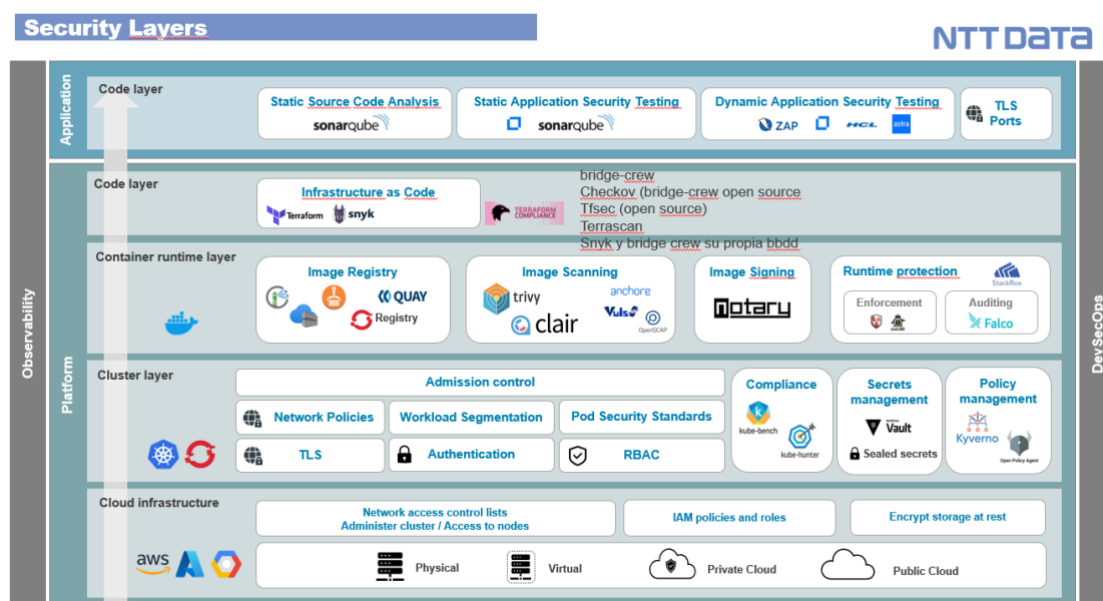


Ilustración 5. Clasificación de herramientas de seguridad existentes

En este proyecto se motiva la importancia de los niveles de seguridad y cuáles son las más empleadas además de una breve comparativa. Y se centrara en el apartado de *Runtime protection* comparando y sugiriendo casos de uso de las herramientas Falco, StackRox y Sysdig Secure.

3. Objetivos

Los objetivos principales que se definieron para este proyecto y que están recogidos en la propuesta de este TFG incluida en el *anexo 1*, son:

- Configurar y desplegar las herramientas de seguridad Sysdig Secure, Falco y StackRox.
- Realizar una comparativa de las herramientas en base a las funcionalidades que ofrecen.
- Crear documentación que permita reproducir lo realizado y configurar las principales buenas prácticas de seguridad sobre las herramientas.

Además de los objetivos principales, se han establecido algunos objetivos secundarios que contribuirán al éxito del proyecto:

- Investigar los niveles de seguridad en Kubernetes
- Identificar y solucionar posibles problemas de compatibilidad entre las herramientas y las plataformas.
- Analizar y comparar las herramientas de seguridad en diferentes niveles
- Detectar las buenas prácticas de las herramientas.
- Identificar las limitaciones y posibles mejoras de las herramientas y en la medida de lo posible proporcionar recomendaciones para su optimización.

Durante la primera fase del proyecto se analizarán estos objetivos con la empresa y se establecerá el alcance del proyecto.

4. Metodología

Este proyecto se va a desarrollar como Trabajo Fin de Grado, asignatura estimada en 12 créditos ECTS lo que supone, aproximadamente, 300 horas de trabajo a desarrollar por, en este caso, una sola persona. La fecha de entrega de este proyecto es el 26 de junio de 2023, por lo que la flexibilidad para la iteración es limitada.

El proyecto se combina con otras asignaturas y trabajo, y el tiempo disponible para su desarrollo no es completo. La disponibilidad para trabajar en este proyecto cambiará de una semana a otra en función de la carga de trabajo externa que se tenga en el momento. Sin embargo, se estima un mínimo de 20 horas semanales. Por tanto, es necesaria una metodología flexible y de respuesta rápida frente a imprevisto.

Teniendo en cuenta las características descritas, se elige Extreme Programming [16] (XP) como metodología para este proyecto.

4.1 Extreme Programming (XP)

Extreme Programming es una metodología de desarrollo de software ágil [16] que enfatiza en el trabajo en equipo, la comunicación, la simplicidad, la retroalimentación y la mejora continua. Está diseñada para entregar software de alta calidad de manera oportuna y rentable, proporcionando un marco para gestionar y responder al cambio durante el proceso de desarrollo. La metodología implica el desarrollo iterativo, la integración continua, pruebas frecuentes y el uso de estándares y buenas prácticas de programación para asegurar la entrega de un software de alta calidad.

Se eligió la metodología ágil XP para el desarrollo de este proyecto debido a su enfoque en la retroalimentación [16], así como en el fomento de buenas prácticas y el uso de diversos elementos considerados importantes. La promoción de la refactorización y la retroalimentación durante todo el proceso de desarrollo es especialmente beneficioso para los objetivos del proyecto, que implica un estudio comparativo de tres herramientas, ya que hasta que no se hayan analizado todas de manera técnica y práctica, no se conocerán las decisiones más acertadas.

La metodología XP se elige también por su enfoque en el uso de buenas prácticas, ya que este es el primer proyecto profesional y se considera una buena oportunidad para adoptar hábitos positivos. Entre los elementos más útiles de XP para este proyecto se encuentran las historias de usuario, las pruebas de aceptación, las estimaciones, la planificación de iteraciones y el registro de comunicaciones entre el cliente y los desarrolladores.

XP se considera especialmente adecuada para proyectos con requisitos imprecisos [16] y altamente cambiantes, así como para aquellos con un alto riesgo técnico, como puede ser el caso

de la falta de experiencia en el entorno de Kubernetes con herramientas desconocidas para este proyecto.

4.2 Aplicación de la tecnología

XP, al igual que la mayoría de las técnicas ágiles [16], se crea para un pequeño equipo de desarrolladores; en consecuencia, la primera diferencia que se observa en el empleo de este enfoque es que este equipo de desarrollo consta de una sola persona. Esto implica la eliminación de reuniones o *stand ups* diarias. En XP, la figura del cliente es altamente significativa; en este caso, NTTDATA ocupará esta posición, y su evaluación e ideas guiarán el progreso del proyecto.

Otro cambio significativo es que las iteraciones ya no serán definidas por un número de días si no por horas. Como se indicó anteriormente, el tiempo disponible para este proyecto es variable. Como resultado, al comienzo de cada iteración, se hará una estimación del tiempo disponible en función de la carga de trabajo externa, y se determinará la duración en la siguiente iteración.

4.3 Seguimiento del desarrollo

Se utilizarán otras herramientas para ayudar a la gestión de proyectos, además de los aspectos empleados por XP para el desarrollo de software.

4.3.1 Trello

Para realizar el seguimiento se ha empleado la herramienta Trello [17]. Se ha seleccionado esta herramienta puesto que estoy familiarizada con ella ya que la he empleado en diversos proyectos durante los estudios académicos y a nivel empresarial.

Trello consiste en una herramienta visual compuesta por un tablero con varias columnas en las que se pueden crear, mover eliminar tarjetas, de esta forma se consigue de forma visual conocer el estado actual del proyecto además de conseguir mayor eficacia en cuanto a tiempo empleado en la herramienta.

En este caso, para abordar un proyecto grande y trabajar de manera iterativa, se ha adoptado el siguiente enfoque. Al comienzo de cada iteración, se desglosan las historias de usuario en tareas más pequeñas. Cada tarea se representa como una tarjeta en el tablero de Trello, y se le asigna una estimación de tiempo y una prioridad. A medida que se avanza en el desarrollo de la iteración y se completan las tareas, se mueven las tarjetas a través de las columnas o estados correspondientes en el tablero de Trello. Estos estados pueden incluir "Por hacer", "En progreso" y "Completado", entre otros, según las necesidades y la estructura del proyecto.

La ilustración 6 muestra el *dashboard* de Trello empleado para el proyecto con las tarjetas de tareas, lo que permite visualizar de manera clara el progreso y el estado actual de cada tarea a

medida que se avanza en la iteración. Esto ayuda a mantener la organización, priorizar el trabajo y realizar un seguimiento efectivo del avance del proyecto.

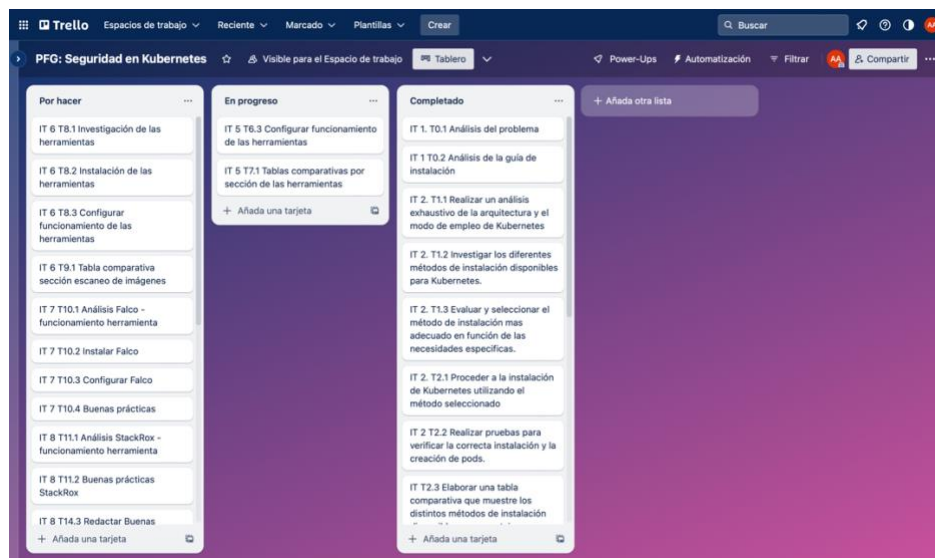


Ilustración 6. Trello 's dashboard del proyecto

4.3.2 Diagrama de trabajo

Para lograr una visualización global más clara del proyecto, se han dividido los tiempos en iteraciones y se ha elaborado un *diagrama de Gantt* utilizando una plantilla de Excel [18]. El diagrama registra el tiempo estimado para cada iteración, permitiendo un análisis detallado de la duración real de cada una. Además, se lleva un registro más minucioso de cada iteración, con el objetivo de comparar los tiempos estimados con los tiempos reales y mejorar las estimaciones futuras a lo largo del proyecto. La ilustración 7 proporciona una estimación del tiempo por iteración durante el período de tiempo asignado al proyecto. Con el objetivo de organizar y gestionar eficientemente el trabajo, se ha dividido el proyecto en varias iteraciones, representadas por diferentes colores en la ilustración 7. Cada iteración está compuesta por diferentes actividades que se llevarán a cabo. En total, contamos con 15 historias de usuario que se encuentran detalladas en el *anexo 2*. Estas historias de usuario se han distribuido en 12 iteraciones, teniendo en cuenta los recursos disponibles y los plazos establecidos. El propósito es abordar gradualmente cada historia de usuario a lo largo de las iteraciones planificadas, asegurando un avance constante y una entrega oportuna. Esta estructura de iteraciones proporciona una visión clara del trabajo a realizar y permite establecer metas alcanzables en cada fase del proyecto. Además, permite ajustar y reasignar recursos según las necesidades y los resultados obtenidos en cada iteración, garantizando así un progreso eficiente y una gestión adecuada del tiempo y los recursos. Dado que el enfoque principal del proyecto es el análisis de herramientas, se considera la opción de realizar las iteraciones de forma simultánea o en secuencia. Sin embargo, se ha decidido llevar a cabo las iteraciones de manera secuencial, siguiendo un orden establecido. De esta manera, se asegura un flujo lógico y organizado en el

desarrollo del proyecto. En el *anexo 3* se encuentra un desglose detallado de cada una de las iteraciones, proporcionando una visualización más detallada de cada una de ellas.

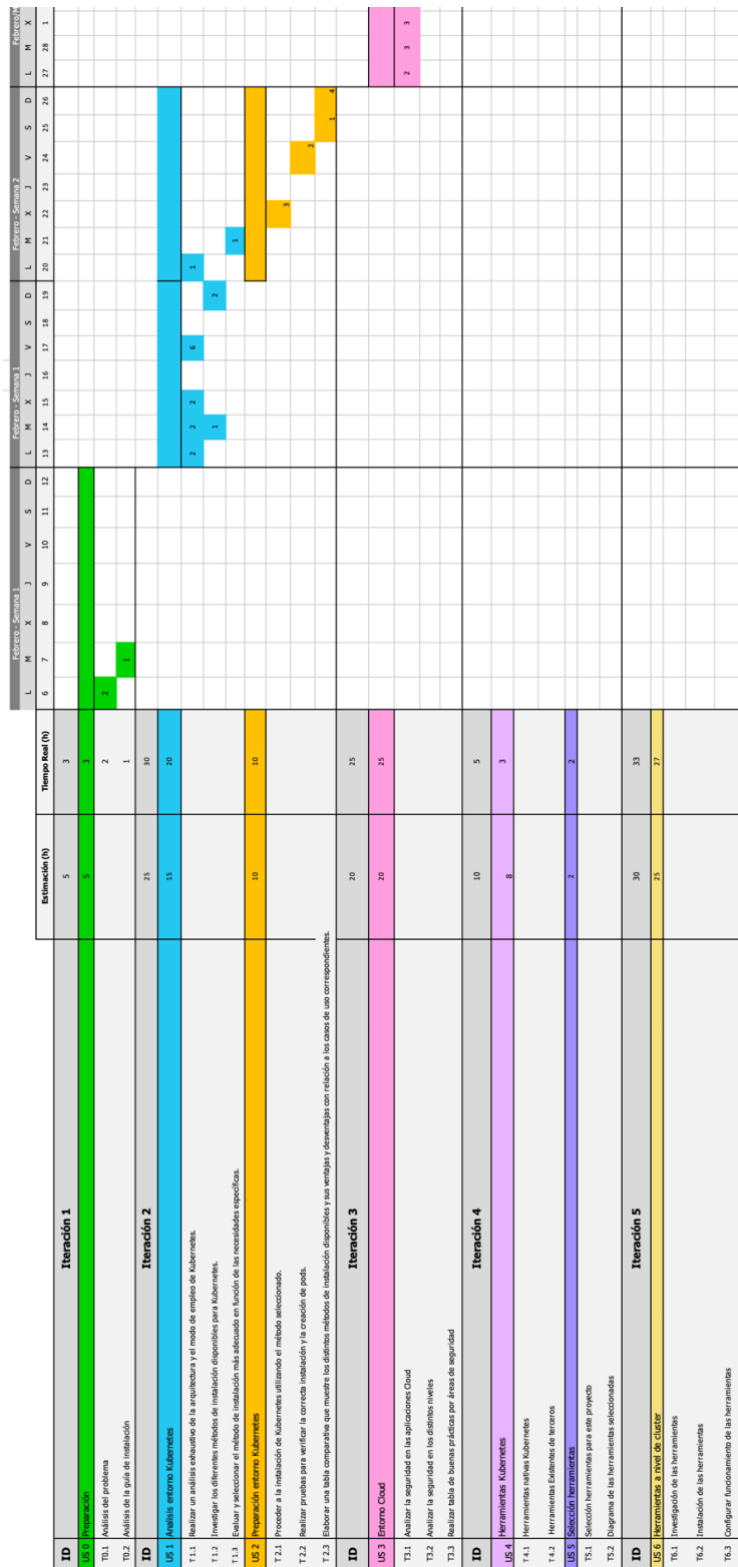


Ilustración 7. Diagrama de Gantt del proyecto

5. Análisis & Implementación

El proyecto tiene como objetivo la fusión de los procesos de análisis e implementación, en virtud de que el proyecto en cuestión se enfoca en el análisis de diversas herramientas. Dado que el enfoque se centra en el análisis de las herramientas y no en el diseño de una solución específica, se ha determinado que el análisis y la implementación deben ir de la mano en este caso particular.

Esta decisión se basa en la necesidad de comprender en profundidad las características y funcionalidades de las herramientas estudiadas, a fin de evaluar su viabilidad y adecuación para el contexto planteado. Asimismo, se busca agilizar el proceso de implementación al integrar el análisis detallado de las herramientas con su posterior puesta en marcha.

De esta manera, se espera obtener un enfoque integral que permita maximizar la eficiencia y los resultados del proyecto, al combinar de manera sinérgica el análisis exhaustivo de las herramientas y su implementación práctica en el contexto específico de estudio.

5.1 Iteración 1: Análisis

5.1.1 Tareas US 0

En esta iteración se realizarán las siguientes tareas de la US0 (Preparación):

01. Análisis del problema
02. Análisis de la guía de instalación

5.1.2 Análisis del problema

La seguridad de Kubernetes depende de diferentes niveles por ello para garantizar la seguridad se necesita proteger:

- A nivel de aplicación
 - Seguridad de código
- A nivel de plataforma:
 - Seguridad de código
 - Seguridad en el nivel en tiempo de ejecución del contenedor (*runtime*)
 - Seguridad a nivel de *clúster*
 - Seguridad de la infraestructura *cloud*

El crecimiento de la adopción de orquestadores, como Kubernetes, ha generado la necesidad de protegerlos, especialmente cuando se despliegan en entornos de producción industriales. La falta de seguridad puede provocar la paralización de la producción y ocasionar pérdidas millonarias. Por tanto, una empresa no puede permitirse una brecha de seguridad tan significativa. Es por

eso por lo que, a través del análisis de herramientas, se busca identificar las mejores soluciones disponibles en el mercado para fortalecer la seguridad de Kubernetes si no, queda muy escueto.

5.1.3 Análisis de la guía de instalación

La adopción por las empresas de la orquestación de contenedores ha llevado a los profesionales técnicos a adquirir conocimientos en este ámbito. La implementación de esta tecnología requiere un entendimiento profundo de la infraestructura y conocimientos técnicos. Por lo tanto, es fundamental que la guía de instalación para el usuario sea lo más sencilla posible, con el fin de facilitar el proceso de instalación y ahorrar tiempo en proyectos futuros.

5.2 Iteración 2 – Investigación & Análisis de Kubernetes

5.2.1 Tareas US 1 y US 2

En esta iteración se realizarán las siguientes tareas de la US1 y US2 (Análisis entorno de Kubernetes, Preparación entorno Kubernetes):

01. Realizar un análisis exhaustivo de la arquitectura y el modo de empleo de Kubernetes.
02. Investigar los diferentes métodos de instalación disponibles para Kubernetes.
03. Evaluar y seleccionar el método de instalación más adecuado en función de las necesidades específicas.
04. Proceder a la instalación de Kubernetes utilizando el método seleccionado.
05. Realizar pruebas para verificar la correcta instalación y la creación de pods.
06. Elaborar una tabla comparativa que muestre los distintos métodos de instalación disponibles, sus ventajas y desventajas con relación a los casos de uso correspondientes.

5.2.2 Desarrollo

La cantidad de herramientas analizar en este proyecto es considerable es por eso por lo que durante las primeras iteraciones se realiza un análisis robusto para poder seleccionar correctamente y realizar las comparativas necesarias durante el proyecto.

Kubernetes proporciona un entorno para coordinar y administrar la ejecución de contenedores en clústeres de servidores [20]. Su principal función es facilitar la implementación, la escalabilidad y la administración de aplicaciones en contenedores. Permite a los desarrolladores agrupar contenedores relacionados en unidades lógicas llamadas *pods* y administrar su implementación, escalado y recuperación automática en caso de fallos. Kubernetes ofrece características avanzadas como la programación automatizada de contenedores, el descubrimiento de servicios, la administración de almacenamiento y la gestión de redes. Además, cuenta con una arquitectura

flexible y modular que permite la integración con diferentes proveedores de infraestructura y herramientas adicionales [20].

En el contexto de este proyecto, se ha determinado que la instalación de Kubernetes se llevará a cabo utilizando el método de instalación a través de Docker con la incorporación de Minikube. Para lograr esto, es necesario realizar la instalación previa de Docker como requisito fundamental.

En el *anexo 5*, se proporciona una guía detallada de instalación que explica paso a paso cómo instalar Docker en el entorno macOS. La inclusión de esta guía se justifica por la necesidad de asegurar una correcta configuración de Docker, que servirá como base para la posterior instalación y funcionamiento de Kubernetes con la ayuda de Minikube.

Minikube es una herramienta de código abierto que permite ejecutar Kubernetes en modo local. Está diseñado para facilitar el desarrollo y la prueba de aplicaciones en entornos de una sola máquina. Minikube crea un clúster de Kubernetes de un solo nodo en una máquina virtual local, lo que permite a los desarrolladores tener una instancia de Kubernetes completamente funcional en su propio equipo. Proporciona una experiencia similar a la de un clúster completo de Kubernetes, lo que permite probar y depurar aplicaciones en un entorno controlado antes de desplegarlas en un entorno de producción. Minikube es fácil de instalar y se puede ejecutar en diferentes sistemas operativos, como Linux, macOS y Windows. Permite a los desarrolladores crear y probar aplicaciones de forma rápida y sencilla, lo que acelera el ciclo de desarrollo y facilita la adopción de Kubernetes en entornos de desarrollo y pruebas.

La ilustración 8 muestra el *dashboard* disponible en Minikube. El *dashboard* proporciona una forma conveniente de obtener una visión completa del estado del trabajo en Kubernetes. En el *dashboard*, se pueden observar diversos gráficos que muestran diferentes características y métricas. El *dashboard* es una interfaz de usuario basada en web de propósito general para clústeres de Kubernetes. Permite a los usuarios gestionar y solucionar problemas tanto de las aplicaciones que se ejecutan en el clúster como del propio clúster en sí. Proporciona información en tiempo real sobre el estado de los pods, servicios, volúmenes, configuraciones y otros recursos en el clúster.

Con el *dashboard*, los usuarios pueden monitorear el rendimiento de las aplicaciones, realizar ajustes en la configuración, escalar los recursos según sea necesario y realizar diversas operaciones de administración de clústeres de forma intuitiva a través de la interfaz gráfica.



Ilustración 8. *Kubernetes dashboard*

La guía de instalación se encuentra en el *anexo 5* y muestra como instalar Minikube.

Al desplegar Kubernetes, se establece un clúster que constituye una parte fundamental de la infraestructura. Un clúster de Kubernetes se compone de una serie de máquinas llamadas nodos de trabajo, los cuales se encargan de ejecutar aplicaciones en contenedores. Es importante destacar que cada clúster cuenta con al menos un nodo de trabajo.

El nodo de trabajo es el anfitrión de los *Pods*, que representan los componentes fundamentales de la carga de trabajo de la aplicación. Además, el plano de control se encarga de gestionar los nodos de trabajo y los *Pods* dentro del clúster. En entornos de producción, el plano de control de Kubernetes suele ejecutarse en varios equipos para garantizar la tolerancia a fallos y la alta disponibilidad del clúster. Además, el clúster en sí puede estar compuesto por múltiples nodos, lo que contribuye a la capacidad de manejar cargas de trabajo de manera eficiente.

La ilustración 9 muestra la arquitectura general de los componentes en Kubernetes, brindando una visión general de cómo se organizan y comunican entre sí. Se pueden observar diferentes elementos clave de la arquitectura de Kubernetes, como el plano de control, que incluye componentes como el API Server, el Controller Manager, el Scheduler y el etcd, que es el almacén de datos distribuido utilizado por Kubernetes. Además, se pueden identificar los nodos del clúster, que son las máquinas físicas o virtuales donde se ejecutan las aplicaciones y los contenedores. Cada nodo tiene un agente llamado *Kubelet*, que se comunica con el plano de control y administra los contenedores en ese nodo. También se observa la presencia de otros componentes, como los servicios, que proporcionan una forma de acceder a las aplicaciones en el clúster, y los volúmenes, que permiten el almacenamiento persistente para los contenedores.

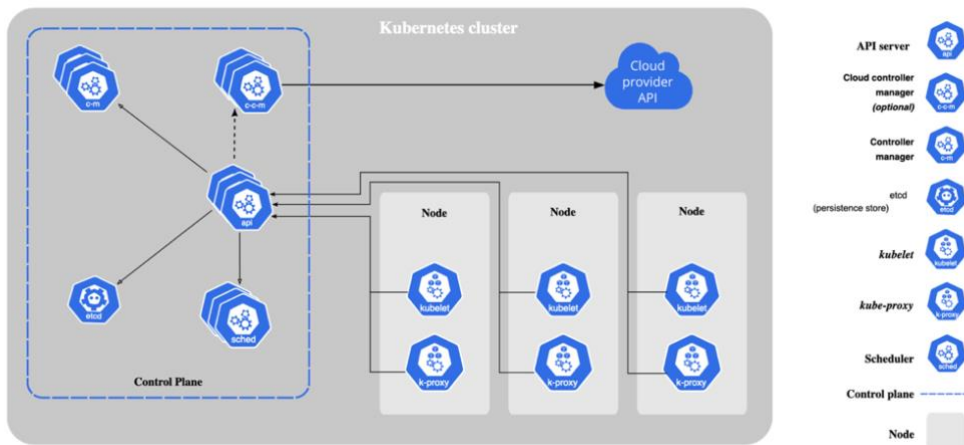


Ilustración 9. Arquitectura Kubernetes

5.2.3 Resultado

En la tabla 1 se presentan y comparan los diferentes métodos de instalación disponibles. Esta tabla permite evaluar y comparar las opciones de instalación disponibles, facilitando la elección del método más adecuado para el entorno y los objetivos del proyecto.

Método de instalación	Ventajas	Desventajas	Casos de uso recomendados
Kubernetes Playground	Fácil de comenzar, no se requiere configuración compleja.	Funcionalidad limitada, no se puede personalizar el entorno o escalar el clúster. Los despliegues no se conservan al cerrar o reiniciar el navegador.	Principiantes o para fines de aprendizaje.
Minikube	Fácil de comenzar y relativamente fácil de instalar.	Funcionalidad limitada, no todas las características de Kubernetes están soportadas. Requiere recursos del equipo local.	Desarrollo local y pruebas iniciales.
Kubeadm	Admite todas las características de Kubernetes. Solución adecuada para casos de uso en producción.	Requiere recursos adicionales, generalmente se ejecuta en múltiples nodos. El proceso de instalación puede ser complicado. Difícil de administrar y mantener.	Implementaciones de producción en entornos locales o en la nube.

Managed Kubernetes (EKS, GKE, AKS)	Solución de nivel de producción respaldada por los proveedores de la nube. Fácil de comenzar y crear clústeres rápidamente. Escalable.	Requiere una cuenta en la nube y puede tener costos asociados. No es mantenible a través de la CLI y puede requerir esfuerzos adicionales para la automatización.	Implementaciones de producción en la nube.
Kubernetes con Terraform	Solución de nivel de producción con características de administración avanzadas. Totalmente mantenible y escalable.	Requiere conocimientos de Terraform. Requiere una cuenta en la nube y puede tener costos asociados. Puede ser un desafío para principiantes.	Implementaciones de producción en la nube con énfasis en la infraestructura como código y la administración avanzada.

Tabla 1. Comparativa de métodos de instalación en Kubernetes

5.3 Iteración 3 – Análisis de las características de seguridad en Kubernetes

5.3.1 Tareas US 3

En esta iteración se realizarán las siguientes tareas de la US3 (Entorno *cloud*):

01. Analizar la seguridad en las aplicaciones *cloud*
02. Analizar la seguridad en los distintos niveles
03. Realizar tabla de buenas prácticas por áreas de seguridad
- 04.

5.3.2 Desarrollo

Con el creciente uso de la nube y la aparición de nuevos perfiles en el ámbito de *DevSecOps*, la seguridad en la nube se ha convertido en un desafío crucial. Para abordar este desafío, la seguridad en la nube se divide en cuatro categorías principales, que se muestran en la ilustración 10 [22]. Estas categorías son:

- Seguridad en *cloud*
- Seguridad en la infraestructura
- Seguridad en los clústeres (imágenes...)
- Seguridad en el código

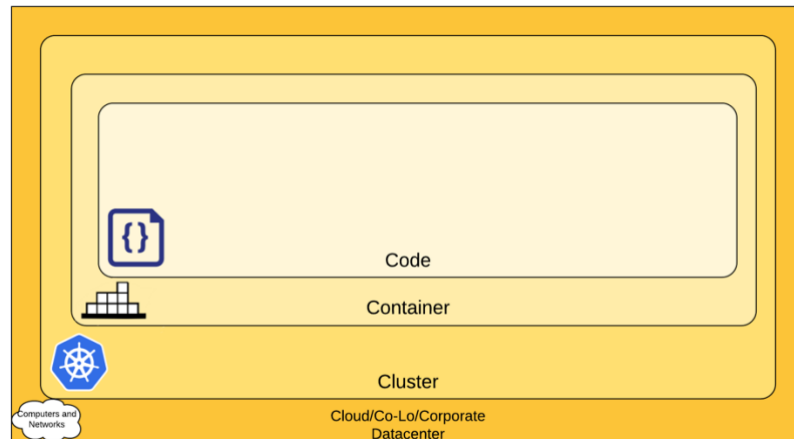


Ilustración 10. Las 4C de la seguridad nativa en la nube

Análisis seguridad en Cloud

La nube juega un papel crucial como una base confiable para un clúster de Kubernetes. Sin embargo, si la capa de la nube presenta vulnerabilidades o está configurada de manera insegura, no se puede garantizar la seguridad de los componentes que se construyen sobre esta base. Cada proveedor de servicios en la nube ofrece recomendaciones de seguridad [23] específicas para asegurar la ejecución segura de cargas de trabajo en su entorno.

Durante mi experiencia en la empresa, he podido observar que diferentes equipos colaboran para identificar problemas de seguridad a lo largo de todo el ciclo de vida del desarrollo. Su objetivo es integrar de manera natural los aspectos relacionados con la seguridad en los flujos de trabajo habituales. Cuanto antes se aborden estos problemas, menor será el riesgo de que el software sea explotado por un atacante. Ahora es común recibir alertas de vulnerabilidades mientras se escribe el código, lo que permite corregirlas antes de avanzar a las siguientes etapas utilizando herramientas de CI/CD. La seguridad del software se integra en todas las etapas del proceso de desarrollo.

Análisis seguridad en la Infraestructura

Además, la seguridad en la infraestructura de Kubernetes es otro aspecto crítico que se aborda de manera rigurosa y efectiva en la empresa. Garantizar la protección de los recursos y datos en un clúster de Kubernetes requiere una atención especial a la seguridad de la infraestructura. Algunas características importantes [23] que se deben tener en cuenta para proteger la infraestructura son:

- **Gestión del acceso:** Es fundamental implementar un control de acceso estricto para restringir quién puede interactuar con la infraestructura de Kubernetes. Se deben establecer políticas y mecanismos de autenticación y autorización sólidos para asegurar que solo usuarios y sistemas autorizados puedan acceder y operar en el clúster.

- **Segmentación de red:** La infraestructura de Kubernetes debe estar adecuadamente segmentada en diferentes redes o subredes, lo que ayuda a reducir el riesgo de ataques y limitar la propagación de posibles compromisos de seguridad. Esta segmentación contribuye a establecer zonas de confianza y aislamiento entre los componentes del clúster.
- **Configuración segura:** Es esencial implementar una configuración segura en todos los componentes de la infraestructura de Kubernetes. Esto incluye asegurar que las políticas de seguridad recomendadas se apliquen correctamente en los nodos, el plano de control, el almacenamiento y otros elementos clave del clúster. Además, es importante mantenerse actualizado con las últimas actualizaciones de seguridad y parches relevantes.
- **Monitoreo y registro:** Se deben implementar mecanismos de monitoreo y registro efectivos para detectar y responder rápidamente a cualquier actividad sospechosa o anómala en la infraestructura de Kubernetes. El seguimiento de registros y la supervisión continua permiten identificar posibles amenazas o incidentes de seguridad y tomar medidas adecuadas para mitigarlos.
- **Gestión de identidad y cifrado:** La gestión de identidades y los mecanismos de cifrado son elementos esenciales para proteger la infraestructura de Kubernetes. Esto incluye la implementación de autenticación sólida y la gestión adecuada de certificados y claves criptográficas. Asimismo, el cifrado de datos en reposo y en tránsito ayuda a salvaguardar la confidencialidad y la integridad de la información sensible.
- **Actualizaciones y evaluaciones regulares:** Es crucial mantener la infraestructura de Kubernetes actualizada con las últimas versiones y parches de seguridad. Además, se deben realizar evaluaciones de seguridad periódicas para identificar posibles vulnerabilidades y realizar mejoras continuas en el entorno.

Análisis de seguridad en los clústeres

A nivel de clúster existen dos áreas de preocupación para asegurar Kubernetes [23]:

- **Asegurar los componentes del clúster que son configurables:** Para garantizar la seguridad de los componentes del clúster de Kubernetes, es crucial asegurar la configuración de Kubernetes, como los archivos de configuración y los parámetros de seguridad, y mantenerlos actualizados. Además, es importante proteger los puntos de acceso al clúster, como el panel de control (Dashboard), el API Server y los nodos, utilizando medidas como la autenticación fuerte, la autorización basada en roles y el cifrado de la comunicación.
- **Asegurar las aplicaciones que se ejecutan en el clúster:** Junto con la seguridad del clúster en sí, también es crucial garantizar la seguridad de las aplicaciones que se

ejecutan en él. Esto implica implementar la incorporación de mecanismos de autenticación y autorización adecuados, el escaneo de imágenes de contenedores en busca de vulnerabilidades y la segmentación adecuada de los recursos y permisos. Además, se deben implementar medidas de detección y respuesta a incidentes para identificar y mitigar posibles amenazas en tiempo real.

Análisis de seguridad en los contenedores e imágenes Docker

La seguridad en los contenedores, como Docker, es un tema crítico que requiere una atención rigurosa y exhaustiva. Aunque los contenedores han ganado popularidad debido a su eficiencia en el despliegue de aplicaciones, también plantean desafíos en términos de seguridad. Por tanto, es importante considerar diversas medidas [23] para garantizar la seguridad de los contenedores.

En primer lugar, se debe seleccionar imágenes base confiables y actualizadas provenientes de fuentes confiables como DockerHub. Estas imágenes deben ser revisadas en busca de posibles vulnerabilidades conocidas. Además, es esencial mantener un proceso de actualización regular y constante para proteger los contenedores contra las últimas amenazas de seguridad y corregir las vulnerabilidades conocidas.

La configuración segura de los contenedores también es fundamental. Se deben aplicar los principios de mínimo privilegio, limitando los permisos y privilegios del contenedor solo a lo necesario. Es importante deshabilitar servicios y características innecesarias, y aplicar políticas adecuadas de acceso y autenticación para prevenir accesos no autorizados.

La comunicación segura entre los contenedores y otros sistemas es crucial. Se recomienda utilizar protocolos seguros como HTTPS y aplicar cifrado para proteger los datos en tránsito. Además, se deben implementar medidas de monitoreo y registro para detectar y responder rápidamente a cualquier actividad sospechosa o incidente de seguridad.

La gestión de vulnerabilidades juega un papel clave en la seguridad de los contenedores. Se deben utilizar herramientas de análisis de vulnerabilidades para identificar posibles brechas de seguridad en las imágenes de los contenedores y en las dependencias utilizadas. También es necesario mantener actualizadas las dependencias con versiones seguras para mitigar riesgos asociados con vulnerabilidades conocidas.

Por último, la educación y concienciación de los desarrolladores y operadores de los contenedores son aspectos esenciales. Se deben proporcionar pautas de seguridad y mejores prácticas, y

fomentar una cultura de seguridad en la que se valore y se tome en serio la seguridad de los contenedores.

En cuanto a la seguridad en las imágenes Docker, es fundamental realizar un análisis exhaustivo antes de su implementación. Esto implica evaluar la imagen en busca de posibles vulnerabilidades conocidas utilizando herramientas y técnicas de análisis automatizado. Además, se deben seguir las mejores prácticas, como utilizar imágenes base confiables y actualizadas, mantener las imágenes actualizadas con correcciones de seguridad y aplicar políticas de control de acceso para restringir modificaciones no autorizadas.

La gestión adecuada de las dependencias utilizadas en las imágenes Docker también es crítica. Se deben evaluar en cuanto a su seguridad y mantenerlas actualizadas con las versiones más seguras disponibles.

Por último, se deben implementar mecanismos de monitoreo y registro para detectar y responder oportunamente a actividades o comportamientos sospechosos relacionados con las imágenes Docker. Esto puede incluir la configuración de alertas y registros de auditoría para detectar intrusiones o intentos de explotación.

Análisis Seguridad en el código

Por último, es importante resaltar la importancia de la seguridad en el código. Aunque el proyecto no se focalice específicamente en este aspecto, resulta fundamental reconocer su existencia y la necesidad de implementar medidas de seguridad a nivel del código [23].

El código de software puede contener vulnerabilidades que podrían ser explotadas por actores maliciosos, lo que podría resultar en brechas de seguridad y comprometer la integridad y confidencialidad de los sistemas. Es esencial adoptar prácticas de desarrollo seguro desde el inicio del ciclo de vida del software. Esto implica implementar técnicas de análisis estático seguro de código (SAST) para identificar posibles problemas de seguridad, como vulnerabilidades conocidas, uso inseguro de funciones o malas prácticas de codificación. El uso de herramientas automatizadas de SAST puede ayudar a identificar y solucionar estas vulnerabilidades de manera eficiente. Las herramientas SAST son un conjunto de tecnologías diseñadas para analizar el código fuente de las aplicaciones para identificar vulnerabilidades de seguridad. Las soluciones SAST analizan una aplicación de "adentro hacia afuera" en un estado de no ejecución. Dichas pruebas escanean antes de que se compile el código. También se conoce como White Box testing. Estas pruebas automatizadas identifican automáticamente vulnerabilidades críticas, tales como desbordamientos de búfer (buffer overflows), inyección SQL (SQL injection), scripts maliciosos

entre otros. Las pruebas de SAST resultan ideales para proporcionar orientación sobre dónde y cómo corregir las vulnerabilidades detectadas en el código fuente.

Además, se debe fomentar la conciencia de seguridad entre los desarrolladores y promover buenas prácticas de codificación segura. Esto implica la adopción de estándares y directrices de codificación segura, como los proporcionados por organizaciones reconocidas como OWASP (Open Web Application Security Project). Se deben educar y capacitar a los desarrolladores sobre los riesgos de seguridad comunes y las mejores prácticas para mitigarlos.

Otro aspecto clave es la gestión de dependencias. El uso de bibliotecas y componentes de terceros es común en el desarrollo de software, pero estas dependencias también pueden contener vulnerabilidades. Es importante mantener un proceso de gestión de dependencias que incluya la evaluación de la seguridad de las bibliotecas utilizadas y la aplicación de actualizaciones de seguridad cuando sea necesario.

La seguridad en el código también implica realizar pruebas exhaustivas, como pruebas de penetración y pruebas de seguridad, para identificar posibles vulnerabilidades antes de implementar el software en entornos de producción. Estas pruebas ayudan a descubrir debilidades y proporcionan oportunidades para fortalecer la seguridad del código.

Análisis de la seguridad reactiva frente a la seguridad proactiva

Es importante distinguir los tipos de seguridad que existen puesto que es necesario integrar ambos para obtener una seguridad más robusta.

Seguridad reactiva

La seguridad reactiva se enfoca en fortalecer las defensas contra los riesgos y tácticas de ataque más frecuentes [24], así como en detectar y resolver los ataques que ya han ocurrido. Las organizaciones utilizan enfoques de seguridad reactiva para hacer frente a los ataques comunes. Aunque estos métodos de seguridad pueden ser eficaces, no deben ser la única opción.

Cuando se realizan pruebas de seguridad o pentesting en un servicio en producción, se está llevando a cabo una estrategia de seguridad reactiva. En este proceso, se busca identificar y solucionar problemas de seguridad antes de que un atacante pueda aprovecharlos.

Seguridad proactiva

Aunque realizar pruebas de seguridad en entornos de producción es siempre recomendable, es aún mejor poder evitar estos problemas de seguridad al implementar los servicios en producción. Esto se puede lograr mediante el uso de mecanismos de seguridad proactiva. La seguridad

proactiva [24] consiste en llevar a cabo acciones antes de lanzar o implementar el servicio en producción. Al utilizar mecanismos de seguridad proactiva, se puede prevenir y evitar vulnerabilidades en los servicios que podrían ser explotadas por atacantes.

Análisis del proceso de seguridad en microservicios

El ciclo de vida en un entorno *cloud* envuelve las tecnologías, prácticas y procesos que ayudan a la resiliencia, manejo, observabilidad de las ejecuciones y seguridad en los entornos *cloud*. En las siguientes secciones llevaremos un análisis detallado de las implicaciones, herramientas, mecanismos y mejores prácticas para integrar la seguridad en el entorno de Kubernetes.

Mecanismos para el desarrollo seguro de los servicios

La seguridad de las aplicaciones nativas de la nube debe abordarse durante todo el ciclo de vida de una aplicación. La fase de desarrollo es la primera etapa de este ciclo, en la cual se crean los artefactos, como la Infraestructura como código, la aplicación y los manifiestos de contenedor, que se utilizarán para implementar y configurar los servicios e infraestructura. Estos artefactos han demostrado ser una fuente de numerosos vectores de ataque que pueden ser explotados durante la ejecución.

A continuación, se presentan una serie de procesos y controles que se deben implementar para reducir drásticamente la superficie de ataque de las aplicaciones [25].

Análisis de seguridad durante el desarrollo

Reforzar la seguridad durante la fase de desarrollo es un componente crítico en el despliegue de aplicaciones. Esto implica que los requisitos de seguridad deben incorporarse en el desarrollo de software y tratarse de la misma manera que cualquier otro requisito de diseño.

Distribución

En esta fase de distribución, se llevan a cabo tareas relacionadas con la verificación de las configuraciones y especificaciones para construir imágenes de contenedor con el mínimo de vulnerabilidades posibles. Esto puede lograrse de forma automática mediante la integración de análisis de seguridad de las imágenes construidas en un flujo de trabajo de CI/CD (Integración Continua/Entrega Continua) y automatizando los despliegues seguros de contenedores Docker. Es necesario incorporar pasos centrados en la seguridad, como escanear las imágenes en busca de amenazas y vectores de ataque, así como validar la integridad de las imágenes.

Escaneo de imágenes

El escaneo de imágenes de contenedor es una de las formas más comunes de proteger las aplicaciones de contenedor a lo largo del ciclo de vida. Es crucial realizar el escaneo en la tubería de CI antes de implementar la imagen en producción. Además, el escaneo continuo de las imágenes de contenedor en ejecución es igualmente importante para identificar vulnerabilidades recién descubiertas.

Tiempo y entorno de ejecución

Esta es una de las partes más desafiantes de proteger, ya que las herramientas de seguridad tradicionales no están diseñadas para monitorear la ejecución de los contenedores, ya que no pueden ver su contenido ni establecer una línea base de su comportamiento esperado.

5.3.3 Resultados

La seguridad en la infraestructura de Kubernetes es de vital importancia para garantizar la integridad y protección de los sistemas. Se debe adoptar un enfoque multidimensional que abarque diversos aspectos clave. A continuación, en la tabla 2, se recopila un resumen de algunas buenas prácticas para garantizar la seguridad de aplicaciones:

Área de Seguridad	Recomendaciones
Infraestructura de Kubernetes	<ul style="list-style-type: none"> • Implementar medidas de control de acceso y autenticación robustos. • Aplicar cifrado de datos en reposo y en tránsito. • Monitorear y registrar la actividad en la infraestructura. • Segmentar la red para reducir el riesgo y limitar la propagación de compromisos
Contenedores	<ul style="list-style-type: none"> ▪ Seleccionar imágenes confiables provenientes de fuentes confiables. ▪ Configurar los contenedores de manera segura con mínimo privilegio. ▪ Proteger la comunicación entre los contenedores y con el exterior mediante la implementación de mecanismos de seguridad adecuados. ▪ Gestionar y aplicar parches de seguridad y actualizaciones de manera regular.

Seguridad en los clústeres de Kubernetes	<ul style="list-style-type: none"> • Asegurar la configuración de los componentes del clúster. • Garantizar la seguridad de las aplicaciones que se ejecutan en el clúster. • Aplicar autenticación, autorización y cifrado de comunicación. • Implementar medidas de detección y respuesta a incidentes.
Seguridad en el código	<ul style="list-style-type: none"> ▪ Adoptar prácticas de desarrollo seguro. ▪ Utilizar herramientas de análisis estático seguro de código (SAST). ▪ Gestionar adecuadamente las dependencias del código y mantenerlas actualizadas. ▪ Fomentar la conciencia de seguridad y promover buenas prácticas de codificación
Imágenes Docker	<ul style="list-style-type: none"> ▪ Realizar un análisis exhaustivo de seguridad antes de la implementación. ▪ Utilizar herramientas automatizadas para identificar y corregir vulnerabilidades. ▪ Utilizar imágenes base confiables y mantenerlas actualizadas. ▪ Seguir las mejores prácticas en la construcción de imágenes Docker. ▪ Gestionar adecuadamente las dependencias de las imágenes. ▪ Implementar mecanismos de monitoreo y registro para detectar actividad sospechosa.

Tabla 2. Características de seguridad en Kubernetes

Al abordar estos aspectos de seguridad desde diferentes perspectivas, se fortalece la seguridad en la infraestructura de Kubernetes, los contenedores, el código y las imágenes Docker, mitigando los riesgos y protegiendo los sistemas y datos de posibles amenazas. En la siguiente iteración se analizan las herramientas existentes para las diferentes áreas de seguridad y se decide cuales emplear para proteger el entorno.

5.4 Iteración 4. Investigación de herramientas de seguridad para Kubernetes

5.4.1 Tareas US 4 y US 5

En esta iteración se realizarán las siguientes tareas de la US 4 y US 5 (Herramientas Kubernetes, Selección herramientas):

02. Herramientas nativas Kubernetes
03. Herramientas existentes de terceros
04. Selección herramientas para este proyecto
05. Diagrama de las herramientas seleccionadas

5.4.2 Desarrollo

Después de un análisis exhaustivo de las diferentes áreas que deben protegerse en este proyecto, se ha decidido enfocarlo en las siguientes áreas propuestas por la empresa:

- Escaneo de imagen
- Cumplimiento normativo
- Protección en tiempo de ejecución
- Gestión de secretos
- Gestión de políticas

Una vez analizadas estas áreas, se investiga acerca de las herramientas nativas de Kubernetes que son específicas para esas áreas.

Se descubre que Kubernetes ofrece configuraciones específicas para la seguridad de algunas áreas como estándares de seguridad, admisiones y control de acceso, políticas de seguridad y secretos en Kubernetes. Por lo tanto, se concluye que Kubernetes no es capaz de cubrir todas las necesidades de seguridad. Por ello, se realiza una búsqueda de herramientas de terceros para implementar en el entorno de Kubernetes y así asegurar la seguridad.

En la investigación, se obtienen las siguientes herramientas. Se analizan las principales herramientas más utilizadas [19], incluyendo aquellas empleadas en una multinacional como NTTDATA. Se obtiene la selección mostrada en la ilustración 11.

5.4.3 Resultados

Es evidente que Kubernetes no puede abarcar todas las áreas de seguridad requeridas para el proyecto. Por lo tanto, es necesario ir más allá e investigar las herramientas disponibles en el mercado actual, a fin de seleccionar las más atractivas y adecuadas.

Durante el proceso de investigación, se obtiene la ilustración 11, donde se consigue un diagrama que muestra las herramientas seleccionadas. Este diagrama representa una visión general de las opciones que se consideran para el proyecto. A medida que se avanza en la investigación, se continúa evaluando y refinando estas opciones. Es importante destacar que la selección final de

las herramientas estará respaldada por un análisis más detallado, considerando también las necesidades específicas del proyecto y la experiencia previa de la empresa.

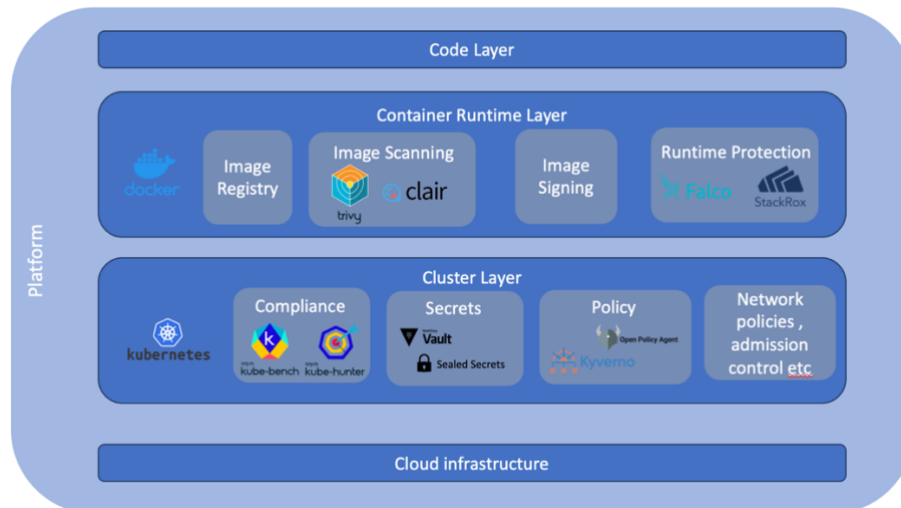


Ilustración 11. Selección de herramientas a analizar

5.5 Iteración 5 – Análisis de las herramientas seleccionadas – *Cluster layer*

5.5.1 Tareas US 6 y US 7

En esta iteración se realizarán las siguientes tareas de la US 6 y US 7 (Herramientas a nivel de clúster, Tabla comparativa - nivel de clúster):

01. Investigación de las herramientas
02. Instalación de las herramientas
03. Configurar funcionamiento de las herramientas
04. Tablas comparativas por sección de las herramientas

5.5.2 Desarrollo

Después de revisar todas las características de Kubernetes, se ha determinado que no todas las configuraciones pueden ser realizadas exclusivamente con esta plataforma. Es por esto por lo que se recurre a herramientas de terceros para cubrir esas necesidades adicionales. Puesto que son diversas herramientas se decide realizar los análisis en dos iteraciones, una iteración por nivel.

A nivel de clúster, se divide la investigación en tres subsecciones: *Compliance*, gestión de secretos y gestión de políticas. Se obtiene una tabla comparativa por sección.

Para cada uno de los ejemplos realizados en la sección *compliance* se emplea la imagen nginx [26].

Compliance

A nivel de *compliance*, **Kube-bench** [27] y **Kube-hunter** [28] son herramientas utilizadas para evaluar la postura de seguridad de los clústeres de Kubernetes, pero tienen diferentes enfoques. **Kube-bench** es una herramienta desarrollada por el *Center for Internet Security* (CIS) para proporcionar un punto de referencia para las mejores prácticas de seguridad de Kubernetes. Consiste en una serie de pruebas que verifican si un clúster de Kubernetes está configurado de acuerdo con el *CIS Kubernetes Benchmark*. Kube-bench puede ayudar a los usuarios a identificar riesgos de seguridad y configuraciones no conformes en sus clústeres, y proporciona orientación sobre cómo remediar esos problemas.

Por otro lado, **Kube-hunter** es una herramienta de código abierto diseñada para analizar los clústeres de Kubernetes en busca de vulnerabilidades de seguridad. Kube-hunter funciona ejecutando una serie de sondas en diferentes componentes del clúster para identificar posibles riesgos de seguridad. Se centra en la identificación de vulnerabilidades y debilidades conocidas que podrían ser explotadas por los atacantes.

En resumen, mientras que Kube-bench se centra en verificar que un clúster de Kubernetes se adhiere a las mejores prácticas de seguridad, Kube-hunter se centra más en encontrar posibles vulnerabilidades y vulnerabilidades. Ambas herramientas son valiosas para evaluar la postura de seguridad de un clúster de Kubernetes, pero sirven para diferentes propósitos y se pueden usar conjuntamente para proporcionar una visión más completa de la seguridad del clúster. Lo ideal es la implementación de las dos herramientas en conjunto.

- Kube-bench

En la ilustración 12, se presentan los logs generados por Kube-bench, una herramienta utilizada para evaluar la seguridad de clústeres de Kubernetes. En los logs, se pueden observar los resultados de las pruebas realizadas, mostrando el estado de cada requisito evaluado. En la parte derecha de los logs, se utiliza el término *[PASS]* para indicar que un requisito ha sido cumplido satisfactoriamente. Estos logs son una representación visual de los resultados obtenidos por Kube-bench, lo que permite a los administradores y equipos de seguridad identificar rápidamente qué requisitos están siendo cumplidos por el clúster y cuáles pueden requerir atención adicional. Esto facilita la tarea de asegurar que el clúster de Kubernetes cumpla con los estándares de seguridad establecidos.

Los pasos para la instalación de Kube-bench se detallan en la guía de instalación, la cual se encuentra en el *anexo 5*.

```

at * minikube
~/.OneDrive /U/4/2/T/k/kube-bench on main > kubectl logs kube-bench-h2zll
[INFO] 1 Control Plane Security Configuration
[INFO] 1.1 Control Plane Node Configuration Files
[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.3 Ensure that the controller manager pod specification file permissions are set to 600 or more restrictive (Automated)
[PASS] 1.1.4 Ensure that the controller manager pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.5 Ensure that the scheduler pod specification file permissions are set to 600 or more restrictive (Automated)
[PASS] 1.1.6 Ensure that the scheduler pod specification file ownership is set to root:root (Automated)

```

Ilustración 12. Ejecución Kube-Bench

En la ilustración 13 se muestran los resultados obtenidos al ejecutar Kube-bench en el entorno correspondiente.

```

== Summary master ==
37 checks PASS
11 checks FAIL
13 checks WARN
0 checks INFO

[INFO] 2 Etcd Node Configuration
[INFO] 2 Etcd Node Configuration
[PASS] 2.1 Ensure that the --cert-file and --key-file arguments are set as appropriate (Automated)
[PASS] 2.2 Ensure that the --client-cert-auth argument is set to true (Automated)
[PASS] 2.3 Ensure that the --auto-tls argument is not set to true (Automated)
[PASS] 2.4 Ensure that the --peer-cert-file and --peer-key-file arguments are set as appropriate (Automated)
[PASS] 2.5 Ensure that the --peer-client-cert-auth argument is set to true (Automated)
[PASS] 2.6 Ensure that the --peer-auto-tls argument is not set to true (Automated)
[PASS] 2.7 Ensure that a unique Certificate Authority is used for etcd (Manual)

== Summary etcd ==
7 checks PASS
0 checks FAIL
0 checks WARN
0 checks INFO

[INFO] 3 Control Plane Configuration
[INFO] 3.1 Authentication and Authorization
[WARN] 3.1.1 Client certificate authentication should not be used for users (Manual)
[INFO] 3.2 Logging
[WARN] 3.2.1 Ensure that a minimal audit policy is created (Manual)
[WARN] 3.2.2 Ensure that the audit policy covers key security concerns (Manual)

```

Ilustración 13. Resultados de la herramienta Kube-Bench

Estos resultados se obtienen en forma de lista con un resumen por sección donde se visualizan los resultados de las pruebas en cuanto a la seguridad que estipula la herramienta. Los resultados son los siguientes:

- **PASS:** Si el test se ha pasado con éxito
- **FAIL:** Si el test ha fallado
- **WARN:** Si hay alguna advertencia
- **INFO:** si hay información al respecto

- Kube-hunter

En la ilustración 14, se pueden observar los logs generados por Kube-hunter, que muestran la información detallada sobre los posibles riesgos encontrados durante el escaneo. Estos logs proporcionan una visión clara de las vulnerabilidades y permiten a los usuarios tomar medidas correctivas para fortalecer la seguridad del clúster.

Los pasos para instalar Kube-hunter se detallan en la guía de instalación, que se encuentra en el *anexo 5*.

certificados y otros datos sensibles. Vault ofrece un conjunto completo de características para la gestión de secretos, incluyendo generación, rotación, revocación y auditoría.

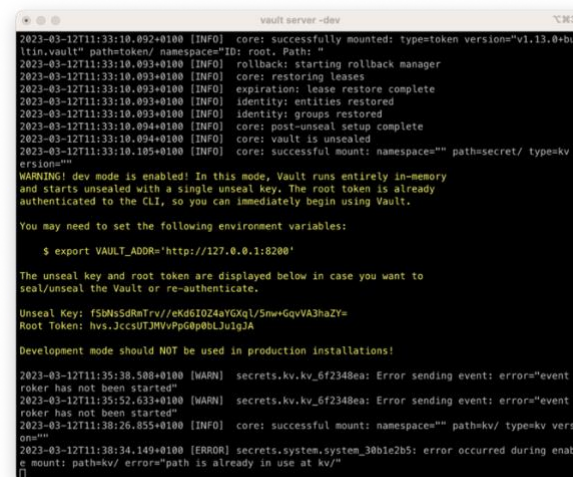
Por otro lado, **Sealed Secrets [30]** es una herramienta que permite el almacenamiento seguro y la gestión de secretos en Kubernetes. Sealed Secrets utiliza la criptografía asimétrica para cifrar los secretos y luego los almacena en el clúster de Kubernetes en forma de objetos sellados. Estos objetos sellados pueden ser descifrados únicamente por un controlador específico, lo que garantiza la seguridad de los secretos almacenados en Kubernetes.

En resumen, Vault es una herramienta más generalizada y versátil para la gestión de secretos en entornos de TI, mientras que Sealed Secrets es específico para Kubernetes y proporciona una forma segura de almacenar secretos en clústeres de Kubernetes.

- Vault

En la ilustración 16, se muestra la ejecución de Vault. Donde se generan los datos del servidor, además de las llaves y tokens necesarios. En la imagen se pueden apreciar los datos generados por el servidor, como las claves y tokens necesarios para el funcionamiento de Vault. Es importante tener en cuenta que estos datos son confidenciales y deben mantenerse seguros. Sin embargo, en este contexto, se presentan solo con fines ilustrativos para comprender el funcionamiento de la herramienta. No se han empleado ni utilizado en ningún entorno real.

Los pasos para la instalación se encuentran en la guía de instalación, en el *anexo 5*.



```
2023-03-12T11:33:10.892+0100 [INFO] core: successfully mounted: typetoken versions=v1.13.0+bu
ltin.vault" path=token/ namespace=""ID: root. Path: "
2023-03-12T11:33:10.893+0100 [INFO] rollback: starting rollback manager
2023-03-12T11:33:10.893+0100 [INFO] core: restoring leases
2023-03-12T11:33:10.893+0100 [INFO] expiration: lease restore complete
2023-03-12T11:33:10.893+0100 [INFO] identity: entities restored
2023-03-12T11:33:10.893+0100 [INFO] identity: groups restored
2023-03-12T11:33:10.894+0100 [INFO] core: post-unseal setup complete
2023-03-12T11:33:10.894+0100 [INFO] core: vault is unsealed
2023-03-12T11:33:10.105+0100 [INFO] core: successful mount: namespace="" path=secret/ type=kv v
ersion=""
WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variables:

$ export VAULT_ADDR='http://127.0.0.1:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: fsmM5dRrtv//ekd6I024yGxqL/5m+GqvVA3haZY=
Root Token: hvs.JccsUTJMVvPpG0p0BLJlqJA

Development mode should NOT be used in production installations!

2023-03-12T11:35:38.508+0100 [WARN] secrets.kv.kv_6f2348ea: Error sending event: error="event b
roker has not been started"
2023-03-12T11:35:52.633+0100 [WARN] secrets.kv.kv_6f2348ea: Error sending event: error="event b
roker has not been started"
2023-03-12T11:38:26.855+0100 [INFO] core: successful mount: namespace="" path=kv/ type=kv versi
on=""
2023-03-12T11:38:34.149+0100 [ERROR] secrets.system.system_38b1e2b5: error occurred during enabl
e mount: path=kv/ error="path is already in use at kv/"
```

Ilustración 16. Ejecución de Vault

Durante la ejecución de Vault se crea un api server desde donde se gestionan los secretos. Es importante a la hora de la instalación guardar la *key* y el *token* generado para autenticación.

A continuación, en la ilustración 17 se muestra el *dashboard* de Vault, una interfaz gráfica desde la cual se pueden gestionar los secretos almacenados. En el *dashboard*, los usuarios tienen la

capacidad de administrar, crear, actualizar y eliminar secretos de manera segura. Esta interfaz proporciona una forma intuitiva y conveniente de interactuar con Vault y garantizar la seguridad de los datos sensibles almacenados en él.

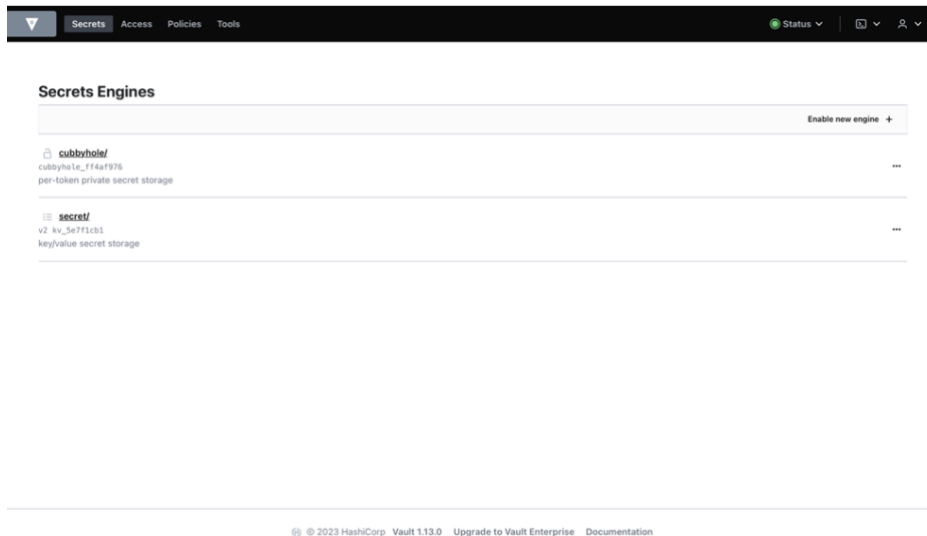


Ilustración 17. Dashboard Vault

A continuación, en la ilustración 18 se presenta un ejemplo de generación de un secreto en Vault utilizando la línea de comandos.

```
andreaalvaromartin@MacBook-Pro-de-ANDREA:~$ vault kv put -mount=secret hello foo=world
the key's current version matches the version specified in the cas parameter. The default is -1.

-mount=<string>
  Specifies the path where the KV backend is mounted. If specified, the next argument will be interpreted as the secret path. If this flag is not specified, the next argument will be interpreted as the combined mount path and secret path, with /data/ automatically appended between KV v2 secrets.

andreaalvaromartin@MacBook-Pro-de-ANDREA:~$ vault kv put -mount=secret hello foo=world
== Secret Path ==
secret/data/hello

==== Metadata =====
Key             Value
-----
created_time    2023-03-12T10:35:38.498674Z
custom_metadata <nil>
deletion_time   n/a
destroyed       false
version         1

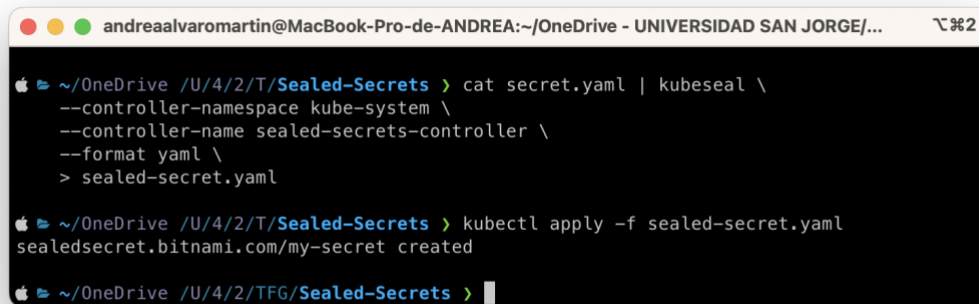
andreaalvaromartin@MacBook-Pro-de-ANDREA:~$ vault kv put -mount=secret hello foo=world excited=yes
== Secret Path ==
secret/data/hello

==== Metadata =====
Key             Value
-----
created_time    2023-03-12T10:35:52.629822Z
custom_metadata <nil>
deletion_time   n/a
destroyed       false
version         2
```

Ilustración 18. Ejemplo Vault

- Sealed Secrets

En la ilustración 19, se muestra el proceso de creación de secretos en Sealed Secrets utilizando archivos YAML. En este caso, se genera un archivo llamado *sealed-secret.yaml* que contiene la configuración del secreto encriptado. Es importante destacar que se pueden crear múltiples archivos YAML para diferentes secretos, cada uno con su propia configuración.



```
andreaalvaromartin@MacBook-Pro-de-ANDREA:~/OneDrive - UNIVERSIDAD SAN JORGE/...
~/OneDrive /U/4/2/T/Sealed-Secrets > cat secret.yaml | kubeseal \
  --controller-namespace kube-system \
  --controller-name sealed-secrets-controller \
  --format yaml \
  > sealed-secret.yaml

~/OneDrive /U/4/2/T/Sealed-Secrets > kubectl apply -f sealed-secret.yaml
sealedsecret.bitnami.com/my-secret created

~/OneDrive /U/4/2/TFG/Sealed-Secrets >
```

Ilustración 19. Ejecución Sealed Secrets

Una vez generados los archivos YAML de los secretos, es necesario aplicarlos al entorno deseado para que se utilicen correctamente. Esto se puede hacer utilizando herramientas como kubectl para aplicar los archivos YAML al clúster de Kubernetes. Al aplicar el archivo *sealed-secret.yaml*, Sealed Secrets se encargará de desencriptar el secreto y almacenarlo de manera segura en el clúster.

Es recomendable revisar el contenido del archivo *sealed-secret.yaml* en el *anexo 4* para comprender la estructura y configuración del secreto encriptado. Este archivo contiene la información necesaria para que el secreto sea utilizado adecuadamente en el entorno de despliegue.

Los pasos para la instalación se encuentran en la guía de instalación, en el *anexo 5*.

Gestión de políticas

En el entorno de gestión de políticas, Open Policy Agent (OPA) y Kyverno son dos herramientas utilizadas en el ecosistema de Kubernetes para implementar y hacer cumplir políticas de seguridad.

Open Policy Agent [31] es un proyecto de código abierto desarrollado por la *Cloud Native Computing Foundation* (CNCF). Proporciona un marco de políticas de acceso y gobernanza para aplicaciones y sistemas. OPA permite definir políticas de manera declarativa utilizando un lenguaje


llamado *Rego*. Estas políticas se evalúan en tiempo de ejecución y se utilizan para tomar decisiones basadas en reglas en diferentes puntos de control, como las solicitudes de admisión, las autorizaciones de acceso y las validaciones de configuración.

Kyverno [32] es otra herramienta de código abierto desarrollada específicamente para Kubernetes. Se centra en la validación y el control de políticas de seguridad en tiempo de ejecución. Kyverno utiliza reglas expresadas en formato YAML para definir políticas de seguridad y gobernanza para los recursos de Kubernetes. Estas políticas se evalúan en tiempo real y pueden aplicarse a los recursos existentes y futuros, asegurando que cumplan con los requisitos de seguridad y gobernanza establecidos.

Ambas herramientas, OPA y Kyverno, son ampliamente utilizadas en entornos de Kubernetes para implementar y hacer cumplir políticas de seguridad y gobernanza. Sin embargo, OPA es un marco de políticas más generalizado y flexible, mientras que Kyverno se centra específicamente en la validación y el control de políticas de seguridad en tiempo de ejecución en Kubernetes. Kyverno esta diseñado específicamente para Kubernetes mientras que OPA como se menciona anteriormente es más general.

- Open Policy Agent

En la ilustración 20 se presenta un ejemplo de política aplicada en Kyverno. Estas políticas se gestionan mediante archivos YAML y se aplican utilizando la línea de comandos.



```
andreaalvaromartin@MacBook-Pro-de-ANDREA:~/OneDrive - UNIVERSIDAD SAN JORGE/USJ/4 CURSO INF INFO/2 SEMESTER/TFG/OPA_GateKeeper
❯ kubectl apply -f all_ns_have_gatekeeper.yml
k8srequiredlabels.constraints.gatekeeper.sh/ns-must-have-gk created
❯ kubectl apply -f bad_NameSpace.yml
Error from server (Forbidden): error when creating "bad_NameSpace.yml": admission webhook "validation.gatekeeper.sh" denied the request: [ns-must-have-gk] you must provide labels: {"gatekeeper"}
❯ kubectl apply -f good_NameSpace.yml
namespace/mynamespace created
```

Ilustración 20. Aplicar política en OPA

Es importante mencionar que las políticas en OPA Gatekeeper permiten establecer reglas y validaciones personalizadas para garantizar el cumplimiento de políticas de seguridad, estándares de codificación, mejores prácticas y otros requisitos específicos del entorno. Estas políticas pueden aplicarse en diferentes objetos de Kubernetes, como *Pods*, *Deployments* o *ConfigMaps*, brindando así un mayor control y seguridad en el clúster.

Se recomienda revisar el archivo YAML en el *anexo 4* para comprender mejor la configuración y las reglas establecidas en esta política aplicada en Kyverno.

Los pasos para la instalación se encuentran en la guía de instalación, en *anexo 5*.

- Kyverno

En la ilustración 21 se presenta un ejemplo de política aplicada en Kyverno. Estas políticas se aplican mediante la línea de comandos utilizando un archivo YAML generado por el usuario.

```

andreaalvaromartin@MacBook-Pro-de-ANDREA:~/OneDrive - UNIVERSIDAD SAN JORGE/USJ/4 CURSO INF INFO/2_SEMESTER/TFG/Kyverno
~/OneDrive /U/4/2_SEMESTER/TFG/Kyverno > kubectl apply -f allowedDeployment.yaml
deployment.apps/nginx-deployment created
  
```

Ilustración 21. Ejemplo aplicación de política

Es importante destacar que Kyverno permite definir políticas personalizadas para aplicar reglas y validaciones en los objetos de Kubernetes. Estas políticas pueden abordar aspectos relacionados con la seguridad, el cumplimiento de estándares, las mejores prácticas y otros requisitos específicos del entorno. Al aplicar estas políticas, Kyverno garantiza que los recursos en el clúster cumplan con las reglas establecidas, proporcionando así una capa adicional de seguridad y control.

Se recomienda revisar detenidamente el archivo YAML en el *anexo 4* para comprender mejor la configuración y las reglas establecidas en esta política aplicada en Kyverno.

Los pasos para la instalación se encuentran en la guía de instalación, en *anexo 5*.

5.5.3 Resultados

Comparativa de herramientas compliance

A continuación, en la tabla 3, se presenta una comparativa de las herramientas Kube-bench y Kube-hunter:

Características	Kube-bench	Kube-hunter
Enfoque	Verifica si la configuración de seguridad del clúster de Kubernetes se implementa de forma segura ejecutando las comprobaciones documentadas en las pruebas del <i>CIS Kubernetes Benchmark</i> .	Busca debilidades de seguridad en clústeres de Kubernetes. La herramienta fue desarrollada para aumentar la conciencia y la visibilidad de los problemas de seguridad en entornos de Kubernetes.
Desarrollado por	Aqua Security	Aqua Security
		Herramienta de escaneo de vulnerabilidades. Se utiliza para buscar

Tipo de herramienta	Herramienta de benchmarking.	vulnerabilidades conocidas en la configuración del clúster.
Tipo de análisis	Estático de la configuración del clúster.	Dinámico del clúster mediante la ejecución de pruebas y escaneos en diferentes componentes del clúster.
Foco en la seguridad	Se enfoca en las configuraciones de seguridad recomendadas por el NCSC de EE. UU.	Busca posibles vulnerabilidades en la infraestructura del clúster de Kubernetes
Método de análisis	Compara las configuraciones del clúster con el <i>benchmarking de CIS</i>	Realiza pruebas y escaneos en diferentes componentes del clúster.
Tipos de vulnerabilidades destacadas	No se centra en vulnerabilidades específicas	Se centra en vulnerabilidades conocidas y debilidades
Formato de salida	Texto plano	Informe detallado en HTML o JSON
Tipo de usuario	Usuarios encargados de configurar y mantener la seguridad del clúster	Usuarios encargados de realizar pruebas de seguridad en el clúster.
Licencia	Código abierto (Apache Licencia 2.0)	Código abierto (Apache Licencia 2.0)
Interfaz	Interfaz de línea de comandos (CLI)	Interfaz de línea de comandos (CLI)
Formato Resultados	Líneas de comandos	Tabla
Facilidad de Uso	+++++	+++++
Facilidad de Instalación	+++++	+++++
Documentación	GitHub	GitHub
Casos de uso	Analizar la seguridad de los clústeres de Kubernetes y los objetos de la API.	Realizar auditorías de seguridad en clúster de Kubernetes.
Integraciones	Integración con herramientas de CI/CD y plataformas de seguridad.	Integración con herramientas de CI/CD y plataformas de seguridad.
Actualizaciones	Actualizaciones regulares y compatibilidad con las últimas versiones de Kubernetes.	Actualizaciones regulares y compatibilidad con las últimas versiones de Kubernetes.

Tabla 3. Comparativa Kube-Bench y Kube-Hunter

En resumen, Kube-bench y Kube-hunter son herramientas complementarias para evaluar la seguridad de los clústeres de Kubernetes. Mientras que Kube-bench se enfoca en verificar si la configuración del clúster cumple con las mejores prácticas de seguridad, Kube-hunter se centra en identificar posibles vulnerabilidades y debilidades en la configuración. Ambas herramientas son de código abierto. Kube-bench cuenta con una interfaz de usuario simple basada en CLI, mientras que Kube-hunter ofrece una interfaz de usuario más visual que muestra una tabla con las vulnerabilidades detectadas y sugiere formas de mitigarlas. En conjunto, estas herramientas proporcionan una evaluación completa de la seguridad de un clúster de Kubernetes, cubriendo tanto la configuración correcta como la identificación de posibles vulnerabilidades. Por ello se recomienda emplear ambas.

Comparativa de herramientas para gestionar secretos: Vault y Sealed Secrets

En la tabla 4, se presenta una comparativa de Sealed Secrets y Vault:

Características	Sealed Secrets	Vault
Tipo de solución	Software Libre	Software libre y empresarial
Tipo de herramienta/ Enfoque	Controlador de Kubernetes	Plataforma de gestión de secretos
Tipo de secretos	Cifrado asimétrico de secretos Kubernetes. Secretos específicos de Kubernetes, como tokens de servicio, credenciales de bases de datos y claves de cifrado	Gestión de secretos de cualquier tipo, incluyendo contraseñas, claves SSH, tokens de API y certificados
Requerimientos de infraestructura	Ninguno adicional, usa Kubernetes como backend	Requiere un servidor de Vault y un backend compatible
Gestión de acceso	Basado en control de acceso de Kubernetes. Acceso a través de API REST o como recurso nativo de Kubernetes.	Ofrece una gran cantidad de opciones de control de acceso. Acceso directo mediante API REST
Escalabilidad	Escala automáticamente con los nodos de Kubernetes. Sealed Secrets es ideal para aplicaciones que se ejecutan en Kubernetes y pueden escalarse fácilmente	Escalabilidad horizontal y vertical. Vault es altamente escalable y puede utilizarse para administrar secretos a gran escala en múltiples plataformas
Forma de almacenamiento	Cifrado en objetos de Kubernetes	Cifrado en repositorio de almacenamiento
Integración con Kubernetes	Está diseñado específicamente para Kubernetes y se integra perfectamente	Es compatible con Kubernetes, pero puede requerir una configuración adicional
Interfaz de usuario	Usa una línea de comando y una API REST	Ofrece una interfaz web y una API REST
Comunidad y soporte	Tiene una comunidad activa y soporte comercial opcional	Tiene una comunidad muy activa y soporte comercial opcional
Encriptación	Utiliza PGP para el cifrado de claves y certificados	Utiliza algoritmos de cifrado AES y RSA
	Específico para Kubernetes y se integra con sus herramientas de gestión de secretos. Sealed Secrets	Puede integrarse con Kubernetes para la gestión de secretos y autenticación de

Integración con Kubernetes	es específico para Kubernetes y se integra con sus herramientas de gestión de secretos, como kubectl y Helm. Los secretos se pueden crear y administrar como objetos de Kubernetes	usuarios. Puede integrarse con Kubernetes para la gestión de secretos y autenticación de usuarios. También incluye una interfaz de línea de comandos y una API REST para acceder y administrar los secretos
Seguridad	Utiliza la seguridad de Kubernetes y PGP para garantizar que los secretos solo puedan ser desbloqueados por destinatarios autorizados.	Ofrece una variedad de características de seguridad, como autenticación de usuario y acceso basado en políticas. Además, incluye características avanzadas de seguridad, como rotación automática de claves y auditoría de registros
Costo	Es una herramienta de código abierto y gratuita	Es una herramienta de código abierto con una versión gratuita y una versión Enterprise con características adicionales
Facilidad de instalación	+++++	++++
Facilidad de uso	++++	+++
Esfuerzo para integrar con Kubernetes.	Poco. K8s Cluster tiene que estar seguro	Moderado, Se requiere Vault y controladores.
Licencia	Apache License 2.0	Mozilla Public License 2.0
Lenguaje	Go	Go
Documentación	GitHub	Hashicorp Vault
Recomendación de uso	Cuando se usan contenedores orquestados por Kubernetes. Los Secrets no se usan en ningún otro lugar. Se requieren otros medios para proporcionar la clave de cifrado de la base de datos.	Cuando se usan secretos que deben compartirse en diferentes plataformas. La integración con K8s es posible a través de proyectos de código abierto.
Interfaz	CLI	CLI - UI

Tabla 4. Comparativa Sealed Secrets y Vault

En resumen, mientras que Sealed Secrets se enfoca específicamente en el almacenamiento seguro de secretos dentro de Kubernetes, Vault ofrece un abanico más amplio de funciones y puede ser utilizado como una solución integral de gestión de secretos para diversas plataformas. La elección entre Sealed Secrets y Vault depende de los requisitos específicos y la complejidad de las necesidades de gestión de secretos en un entorno de Kubernetes.

Comparativa Herramientas para gestionar políticas: OPA y Kyverno

Kyverno y Open Policy Agent (OPA) son dos herramientas populares de política de seguridad de Kubernetes que se utilizan para aplicar políticas de seguridad y reglas de cumplimiento en entornos de Kubernetes. Seguidamente, se presenta la tabla 5 correspondiente a una comparativa detallada y completa de Kyverno y OPA.

Características	Kyverno	Open Policy Agent (OPA) GateKeeper
Validación	Si	Si
Mutación	Si (versión Beta)	Si
Mutación para recursos existentes	Si	No
Generación	Si	No
Limpieza	Si	No
Verificación de Imagen	Si	Si (Via extensiones)
Verificación Manifiesto (Sigstore)	Si	No
Registro de Imagen	Si	Si (Via extensiones)
Extensiones	No	Si (Version Beta)
Políticas como fuentes nativas	Si	Si
Exposición de Metricas	Si	Si
Open API esquema de validación (kubectrl explain)	Si	No
Alta disponibilidad	Si	Si
Búsqueda de objetos de API	Si	Si
CLI with test	Si	Si
Policy audit ability	Si	Si
Self-service reports	Si	No
Self-service exceptions	Si	No
Licencia	Apache 2.0	Apache 2.0
Facilidad de instalación	++++	+++
Facilidad de uso	+	++
Programación requerida	Si	Si
Uso fuera de Kubernetes	No	Si
Madurez de la documentación	●	◐
Biblioteca de ejemplos de políticas	Si	Si
Número de ejemplos de políticas de la comunidad	46	265

Tabla 5. Comparativa Kyverno y Open Policy Agent

Con conclusión del análisis de Kyverno y Open Policy Agent – GateKeeper se obtiene la tabla 6.

	Ventajas	Desventajas
GateKeeper	<ul style="list-style-type: none"> Capaz de expresar políticas muy complejas 	<ul style="list-style-type: none"> Requiere un lenguaje de programación. Por tanto, implica conocimientos técnicos de <i>Rego</i>.

	<ul style="list-style-type: none"> ○ Más establecido en la comunidad 	<ul style="list-style-type: none"> ○ La capacidad de mutación está en desarrollo. ○ Su utilidad se limita a casos de uso predominantemente de validación. ○ La política es compleja, a menudo larga y requiere múltiples documentos para implementarla.
Kyverno	<ul style="list-style-type: none"> ○ Simplicidad de la expresión de las políticas que permite la estructura de Kubernetes. ○ Habilidades de mutación sólidas. ○ La verificación de imagen integrada reduce las complejidades externas. ○ Generación, sincronización, limpieza y otras habilidades son únicas y permiten nuevos casos de uso. ○ Tiempo rápido para valorar y alto grado de flexibilidad. 	<ul style="list-style-type: none"> ○ No es posible definir una política muy compleja ya que no permite usar ningún lenguaje de programación. ○ Muy reciente, todavía no está tan integrado en las organizaciones.

Tabla 6. Comparativa OPA GateKeeper y Kyverno ventajas y desventajas

En resumen, OPA Gatekeeper es una herramienta más flexible y potente para hacer cumplir políticas en Kubernetes, ya que se basa en el potente motor de políticas de OPA. Kyverno, por su parte, es una solución más ligera y nativa de Kubernetes que proporciona una forma sencilla de definir y aplicar políticas de cumplimiento. La elección entre OPA Gatekeeper y Kyverno dependerá de las necesidades específicas y la complejidad de las políticas de cumplimiento en el entorno de Kubernetes.

5.6 Iteración 6. Análisis de las herramientas en tiempo de ejecución

5.6.1 Tareas US 8 y US 9

En esta iteración se realizarán las siguientes tareas de la US 8 y US 9 (Herramientas a nivel de tiempo de ejecución):

01. Investigación de las herramientas
02. Instalación de las herramientas
03. Configurar funcionamiento de las herramientas
04. Tabla comparativa sección escaneo de imágenes

5.6.2 Desarrollo

En esta iteración se analizan las herramientas a nivel de tiempo de ejecución. La investigación se divide en dos subsecciones: escaneo de imagen y protection en tiempo de ejecución. Se analizan

dichas herramientas además de realizar la instalación. A continuación, se obtiene una tabla comparativa por sección. Cabe destacar la sección de registro de imágenes y las firmas de imágenes puesto que son áreas fundamentales para la protección de los contenedores en Kubernetes se mencionarán y explicarán.

Registro de contenedores

Un registro de contenedores [33] es un repositorio o colección de repositorios utilizado para almacenar y acceder a imágenes de contenedores. Estos registros son utilizados en el desarrollo de aplicaciones basadas en contenedores, a menudo como parte de los procesos de *DevOps*. Pueden conectarse directamente a plataformas de orquestación de contenedores como Docker y Kubernetes. El registro de imágenes empleado en la empresa NTTDATA es Harbor [34]. Por ello, se procederá a comentar de forma general dicha herramienta.

Harbor

Harbor [34] es un registro de código abierto que ofrece la flexibilidad de ser instalado en prácticamente cualquier entorno, pero se destaca especialmente por su compatibilidad con Kubernetes. A diferencia de algunos servicios que integran estrechamente sus registros con sus propias soluciones, la versatilidad de Harbor le permite adaptarse a diferentes contextos. Es compatible con la mayoría de los servicios en la nube y plataformas de Integración Continua y Entrega Continua (CI/CD), lo que lo convierte en una opción adaptable. Además, también ofrece una solución sólida para implementaciones locales *on-premises*.

Firmas de imágenes

Firmar código, ejecutables o scripts y asegurarse de instalar solo artefactos de software confiables se considera una práctica recomendada para garantizar la autenticidad e integridad. El concepto de firmar artefactos de software no es nuevo, hay una gran cantidad de proveedores de soluciones y servicios en el mercado y, básicamente, cada organización tiene sus propias formas de manejar la firma y la aplicación de confianza en los artefactos de software. Sin embargo, la herramienta más conocida para la firma de imágenes es Notary.

Notary

Notary [35] es una herramienta de código abierto basada en la especificación TUF [36] que permite publicar y administrar colecciones de contenido confiable. Los editores tienen la capacidad de firmar digitalmente estas colecciones, mientras que los consumidores pueden verificar la integridad y procedencia del contenido. Todo esto se logra a través de una interfaz intuitiva para la gestión de claves y firmas, que facilita la creación de colecciones firmadas y la configuración de editores de confianza.

Gracias a Notary, cualquier persona puede generar confianza en colecciones de datos arbitrarias. Utilizando el Marco de Actualización (TUF) como marco de seguridad subyacente, Notary se

encarga de todas las operaciones necesarias para crear, administrar y distribuir los metadatos esenciales para garantizar la integridad y actualidad del contenido. Además, realiza la firma de imágenes utilizando la jerarquía de roles y claves establecida por TUF [35].

Escaneo de imagen

El escaneo de imágenes en Kubernetes se refiere al proceso de analizar y evaluar la seguridad de las imágenes de contenedores que se utilizan en un entorno de Kubernetes. Esto implica detectar y prevenir posibles vulnerabilidades, configuraciones inseguras o *malware* presentes en las imágenes antes de que se desplieguen en un clúster de Kubernetes.

El escaneo de imágenes se realiza generalmente mediante herramientas automatizadas que examinan el contenido de las imágenes en busca de posibles problemas de seguridad. Estas herramientas verifican la presencia de paquetes, bibliotecas o dependencias desactualizadas o conocidas por tener vulnerabilidades conocidas. También pueden evaluar la configuración y el cumplimiento de las políticas de seguridad definidas. En este caso se han seleccionado Trivy y Clair puesto que son las empleadas en la empresa.

El objetivo del escaneo de imágenes en Kubernetes es garantizar que solo se utilicen imágenes de contenedores confiables y seguras en un entorno de producción. Al identificar y solucionar problemas de seguridad en las imágenes antes de su implementación, se reduce el riesgo de exposición a posibles amenazas y se mejora la postura general de seguridad del clúster de Kubernetes.

Trivy

Trivy [37] es una herramienta de escaneo de vulnerabilidades diseñada específicamente para entornos de contenedores, incluido Kubernetes. Proporciona una forma automatizada de analizar las imágenes de contenedores en busca de posibles vulnerabilidades y configuraciones inseguras.

Trivy utiliza una amplia base de datos de vulnerabilidades conocidas para comparar el contenido de las imágenes y detectar posibles problemas de seguridad. Examina las bibliotecas, dependencias y paquetes utilizados en la imagen para identificar cualquier elemento que pueda tener vulnerabilidades conocidas. La herramienta es fácil de usar y se puede integrar en los flujos de trabajo de desarrollo y despliegue de Kubernetes. Puede ejecutarse de manera autónoma o como parte de una tubería de CI/CD para analizar las imágenes de contenedor antes de que se desplieguen en un clúster de Kubernetes.

Trivy proporciona información detallada sobre las vulnerabilidades encontradas, incluyendo su gravedad y recomendaciones para mitigar los riesgos. Esto permite a los administradores de Kubernetes tomar medidas correctivas y actualizar las imágenes de contenedor con versiones más seguras.

En la ilustración 22, se presentan los resultados de un ejemplo [38] en el cual se ha aplicado la herramienta Trivy. En dicha ilustración, se muestra una tabla resumen que proporciona información sobre los diferentes *namespaces* y las vulnerabilidades, configuraciones y secretos asociados a ellos. Estos elementos se han categorizado en función de la importancia del riesgo, utilizando las etiquetas *High*, *Medium* y *Low* (H, M, L).

```

Summary Report for minikube

Workload Assessment
-----
| Namespace | Resource | Vulnerabilities | Misconfigurations | Secrets |
|-----|-----|-----|-----|-----|
| kube-system | Pod/etcd-minikube | 20 C 8 H | 1 C 3 H 7 M | 0 C 0 H 0 M 0 L 0 U |
| kube-system | Pod/kube-scheduler-minikube | 1 C | 1 C 3 H 7 M | 0 C 0 H 0 M 0 L 0 U |
| kube-system | DaemonSet/kube-proxy | 12 C 9 H 23 M | 2 C 3 H 10 M | 0 C 0 H 0 M 0 L 0 U |
| kube-system | Pod/kube-apiserver-minikube | 1 C 1 H 1 M | 1 C 3 H 6 M | 0 C 0 H 0 M 0 L 0 U |
| kube-system | Pod/kube-controller-manager-minikube | 1 C 1 H 1 M | 1 C 3 H 6 M | 0 C 0 H 0 M 0 L 0 U |
| kube-system | Deployment/coredns | 1 C 3 H 2 M | 2 C 5 H 10 M | 0 C 0 H 0 M 0 L 0 U |
| kube-system | Pod/storage-provisioner | 1 C 10 H 3 M | 1 C 3 H 10 M | 0 C 0 H 0 M 0 L 0 U |
| kube-system | ConfigMap/extension-apiserver-authentication | 1 C | 1 C | 0 C 0 H 0 M 0 L 0 U |

Severity: C=CRITICAL H=HIGH M=MEDIUM L=LOW U=UNKNOWN

RBAC Assessment
-----
| Namespace | Resource | RBAC Assessment |
|-----|-----|-----|
| kube-system | Role/system:leader-locking-kube-scheduler | 1 C |
| kube-system | Role/system:persistent-volume-provisioner | 2 C |
| kube-system | Role/system:controller:cloud-provider | 1 C |
| kube-system | Role/system:controller:bootstrap-signer | 1 C |
| kube-system | Role/system:controller:token-cleaner | 1 C |
| kube-system | Role/system:leader-locking-kube-controller-manager | 1 C |

Severity: C=CRITICAL H=HIGH M=MEDIUM L=LOW U=UNKNOWN

```

Ilustración 22. Resultados Trivy

La tabla resumen permite tener una visión clara y organizada de las vulnerabilidades y configuraciones problemáticas encontradas en cada *namespace*. Las categorías de riesgo (High, Medium, Low) indican el nivel de gravedad asociado a cada problema identificado. Esto ayuda a priorizar las acciones de mitigación y corregir las vulnerabilidades más críticas primero.

La utilización de la herramienta Trivy en este ejemplo ha permitido realizar un análisis exhaustivo de los diferentes *namespaces*, identificando y clasificando las vulnerabilidades, configuraciones y secretos encontrados. Esto facilita la tarea de gestionar y remediar los problemas de seguridad.

Clair

Clair [39] es una herramienta de escaneo de seguridad para contenedores, similar a Trivy. Está diseñada específicamente para analizar imágenes de contenedores en busca de posibles vulnerabilidades y riesgos de seguridad.

Al igual que Trivy, Clair utiliza una base de datos actualizada de vulnerabilidades conocidas para comparar el contenido de las imágenes de contenedor y detectar cualquier componente que pueda tener vulnerabilidades conocidas. La herramienta analiza las bibliotecas, dependencias y paquetes utilizados en las imágenes y proporciona información detallada sobre las vulnerabilidades encontradas.

Clair se integra bien con los flujos de trabajo de desarrollo y despliegue de contenedores, incluido Kubernetes. Puede ser utilizado como parte de un proceso de construcción y despliegue de imágenes de contenedor, proporcionando información en tiempo real sobre las vulnerabilidades detectadas.

Una de las características distintivas de Clair es su capacidad para realizar un análisis estático de imágenes de contenedor almacenadas en registros de contenedores, como *Docker Registry*. Esto permite escanear y monitorear imágenes en repositorios públicos o privados, brindando visibilidad sobre las vulnerabilidades presentes en las imágenes utilizadas en el entorno de Kubernetes.

Runtime protection

A nivel del tiempo de ejecución en Kubernetes es necesario realizar una implementación de medidas y controles de seguridad para garantizar la protección continua de las aplicaciones y los contenedores mientras se están ejecutando en un clúster de Kubernetes.

En un entorno de Kubernetes, la protección en tiempo de ejecución aborda los posibles riesgos y amenazas que pueden surgir durante la ejecución de los contenedores. Esto incluye la detección y prevención de comportamientos maliciosos, la mitigación de ataques y la supervisión constante de la integridad y el rendimiento de las aplicaciones.

Para ello se han seleccionado las siguientes herramientas:

- Falco
- StackRox
- Sysdig Secure

La siguiente iteración, se enfocará en explorar y trabajar de manera más detallada con las herramientas mencionadas anteriormente, buscando aprovechar al máximo sus funcionalidades y beneficios.

5.6.3 Resultados

Comparativas de las herramientas para la gestión de secretos: Trivy y Clair

Clair y Trivy son dos herramientas de escaneo de seguridad y aunque comparten un propósito similar, existen algunas diferencias entre ellas:

Características	Clair	Trivy
Soporte de sistemas Operativos	Linux	Linux, macOS, Windows
Escaneo de vulnerabilidades	Si	Si
Escaneo de paquetes de bibliotecas	Si	Si
Escaneo de archivos de configuración	No	Si
Escaneo de imagen base	Si	Si
Integración con herramientas de CI/CD	Si	Si
Escaneo en tiempo real	No	No
Integración con herramientas de orquestación de seguridad	Si	Si
Facilidad de instalación	+++	++++
Facilidad de Uso	+++	+++
Documentación	GitHub	Aqua Security / GitHub
Tipo de escaneo	Análisis estático	Análisis estático y dinámico
Base de datos	Red Hat	Varias fuentes de datos de vulnerabilidades
Integración	Flujos de trabajo de desarrollo y despliegue de contenedores	Herramientas de CI/CD, también se puede utilizar como biblioteca
Soporte de plataformas	Docker, Kubernetes y otras plataformas de contenedores	Docker, Kubernetes, OCI, <i>Dockerfile</i>
Flexibilidad	Análisis de imágenes almacenadas en repositorios públicos o privados	Análisis de imágenes locales y en repositorios
Actualizaciones	Actualizaciones frecuentes y mantenimiento activo	Actualizaciones regulares y mejoras continuas
Usabilidad	Interfaz de línea de comandos (CLI) y API disponible	Interfaz de línea de comandos (CLI) y biblioteca utilizada en el código
Comunidad	Amplia comunidad de usuarios y contribuidores	Comunidad activa y creciente

Tabla 7. Comparativa Clair y Trivy

5.7 Iteración 7 – Análisis Falco

5.7.1 Tareas US 10 y US 14

En esta iteración se realizarán las siguientes tareas de la US 10 y US 14 (Análisis de la herramienta Falco, Guía de instalación):

01. *Análisis Falco – funcionamiento herramienta*
02. *Instalar Falco*
03. *Redactar Instalación Falco – Guía de instalación*
04. *Configurar Falco*
05. *Buenas prácticas Falco*
06. *Redactar buenas prácticas Falco – Guía de instalación*

5.7.2 Desarrollo

Durante esta iteración, se realiza el análisis de la herramienta Falco, seguido de su instalación y configuración para explorar las diferentes opciones que ofrece. A partir de estas configuraciones, se obtuvieron una serie de buenas prácticas para lograr una mayor eficacia en la instalación y configuración de la herramienta.

Falco

Falco [40] es una solución de seguridad de código abierto diseñada específicamente para entornos de Kubernetes. Su función principal es detectar comportamientos anómalos y generar alertas en tiempo real para proteger los clústeres de Kubernetes. A continuación, se presentan algunas características destacables de Falco en términos de seguridad:

- **Detección de comportamientos anómalos:** Falco utiliza reglas y políticas predefinidas para identificar actividades sospechosas o maliciosas en tiempo real. Puede monitorear eventos a nivel del sistema operativo y del contenedor, lo que le permite detectar actividades inusuales, como la ejecución de comandos de *shell* sospechosos, la lectura o escritura de archivos inapropiados y el acceso no autorizado a recursos del clúster.
- **Motor de detección basado en reglas:** Falco cuenta con un motor de detección potente basado en reglas, que permite a los usuarios definir y personalizar sus propias reglas de seguridad en archivos YAML. Esto brinda flexibilidad para adaptarse a las necesidades específicas de cada entorno de Kubernetes. En Falco, estas reglas también permiten generar listas macro que facilitan la adición de comandos maliciosos.
- **Integración con el kernel del sistema operativo:** Falco aprovecha las capacidades de trazado del kernel del sistema operativo subyacente para capturar eventos a nivel del



sistema y del contenedor. Esto le permite monitorear y detectar actividades anómalas en tiempo real sin necesidad de realizar modificaciones en los contenedores en ejecución.

- **Alertas en tiempo real:** Cuando Falco detecta una actividad sospechosa, genera alertas inmediatas. Estas alertas pueden enviarse a múltiples destinos, como sistemas de registro (por ejemplo, Elasticsearch), sistemas de gestión de eventos o canales de notificación como Slack, para que los equipos de seguridad puedan responder rápidamente a posibles amenazas.
- **Integración con herramientas de respuesta a incidentes:** Falco se puede integrar con herramientas de respuesta a incidentes y orquestación, lo que permite una respuesta automatizada a eventos de seguridad. Por ejemplo, se puede configurar para bloquear automáticamente un contenedor comprometido o desencadenar una serie de acciones para mitigar una posible amenaza.
- **Comunidad activa y actualizaciones regulares:** Falco es un proyecto de código abierto respaldado por una comunidad activa de desarrolladores y usuarios. Esto garantiza que se realicen mejoras y actualizaciones regulares, lo que permite mantenerse al día con las últimas amenazas y técnicas de seguridad.

Una de las características interesantes de Falco son sus alertas en tiempo real. Cuando se viola una regla de Falco, se activa una alerta. Las alertas se pueden enviar a múltiples canales compatibles:

- Salida estándar (por ejemplo, *stdout*, registros de Docker, registros de kubectl)
- *syslog*
- Archivos
- Entrada a un programa (por ejemplo, correo electrónico)

En la ilustración 23, se muestra un ejemplo de alerta en Falco

```
root@host:~# tail /var/log/syslog | grep "host falco"
Jun 21 14:15:45 p-egab87cmffwj-host falco[4987]: 14:15:45.604313276: Error File below /etc opened for writing (user=root user_loginuid=1 command=event-generator run syscall --loop pid=6438 parent=event-generator pcmdline=event-generator run syscall --loop file=/etc/created-by-event-generator program=event-generator gparent=<NA> gparent=<NA> container_id=59143c670750 image=falcosecurity/event-generator)
Jun 21 14:15:45 p-egab87cmffwj-host falco[4987]: 14:15:45.840786240: Notice Database-related program spawned process other than itself (user=root user_loginuid=1 program=ls pid=14779 parent=mysqlld container_id=59143c670750 image=falcosecurity/event-generator)
Jun 21 14:15:45 p-egab87cmffwj-host falco[4987]: 14:15:45.943296444: Error Directory below known binary directory created (user=root user_loginuid=1 command=event-generator run syscall --loop pid=6438 directory=/bin/directory-created-by-event-generator container_id=59143c670750 image=falcosecurity/event-generator)
Jun 21 14:15:52 p-egab87cmffwj-host falco: 14:15:52.080345448: Warning Sensitive file opened for reading by trusted program after startup (user=root user_loginuid=1 command=httpd --loglevel info run ^syscall.ReadSensitiveFileOntusted$ --sleep 6s pid=14780 parent=httpd file=/etc/shadow parent=httpd gparent=event-generator container_id=59143c670750 image=falcosecurity/event-generator)
Jun 21 14:15:52 p-egab87cmffwj-host falco: 14:15:52.317578440: Notice Unexpected setuid call by non-sudo, non-root program (user=daemon user_loginuid=1 cur uid=2 parent=child command=child --loglevel info run ^syscall.NonSudoSetuid)
```

Ilustración 23. Formato alerta en Falco

Finalmente, existen otras formas de generar alertas de Falco. Son especialmente útiles cuando los registros serán consumidos externamente:

- Métodos HTTP[s]
- Clientes gRPC

Para utilizar cualquiera de las opciones mencionadas anteriormente, se debe habilitar el formato JSON.

Falco es una herramienta que brinda una capa adicional de seguridad para entornos de Kubernetes al monitorear y detectar comportamientos anómalos en tiempo real. Su capacidad de personalización, integración y generación de alertas lo convierten en una herramienta valiosa para fortalecer la seguridad en los clústeres de Kubernetes. Es importante tener en cuenta que para aprovechar al máximo Falco, se requieren conocimientos previos de programación y de la estructura de archivos YAML, ya que la personalización de reglas se realiza a través de archivos YAML. Sin embargo, una vez dominados estos conceptos, Falco puede ser una poderosa herramienta en manos de los profesionales de seguridad para detectar y responder a amenazas en entornos de Kubernetes. Además, Falco utiliza una serie de listas predefinidas para mejorar la detección de comportamientos anómalos y maliciosos en entornos de Kubernetes. Estas listas incluyen comandos sospechosos, archivos o directorios sensibles, puertos o servicios no autorizados, direcciones IP o dominios maliciosos conocidos, y patrones de comportamiento sospechoso. Estas listas son actualizadas y mantenidas por la comunidad de seguridad y se pueden personalizar según las necesidades específicas del entorno.

Falco es una herramienta de terceros que se integra fácilmente con Kubernetes. Además, permite integrar alertas en servicios en la nube y en plataformas como AWS, lo cual resulta especialmente útil para grandes proyectos. En NTTDATA, se emplea Falco en casos de este tipo.

La integración con herramientas como ErgoCD permite enlazar y conectar Falco para registrar y analizar lo que está ocurriendo en el entorno.

5.7.1 Resultados

La instalación de Falco se encuentra en la **guía de instalación en el *anexo 5***. Además de las buenas prácticas para la gestión de Falco.

5.8 Iteración 8 – Análisis StackRox

5.8.1 Tareas US 11 y US 14

En esta iteración se realizarán las siguientes tareas de la US 11 y US 14 (Análisis StackRox, Guía de instalación):

- 01. Análisis StackRox – funcionamiento herramienta*
- 02. Instalar StackRox*
- 03. Redactar Instalación StackRox – Guía de instalación*
- 04. Configurar StackRox*

~~05.- Buenas prácticas StackRox~~

~~06.- Redactar Buenas prácticas StackRox — Guía de instalación~~

5.8.2 Desarrollo

Durante esta iteración, se realizó el análisis de la herramienta StackRox, se observa que no dispone de versión gratuita. Por tanto, se modifican las tareas en la iteración 8 US8. Se analiza las características más destacables para poder realizar la comparativas además de ver videos explicativos del funcionamiento e instalación para proporcionar la información necesaria a la hora de comparar.

StackRox

StackRox [41] es una tecnología de seguridad de contenedores que ofrece una protección continua para la configuración de Kubernetes. Como parte de Red Hat, esta solución destaca por varias características clave.

En primer lugar, StackRox se enfoca en proteger las aplicaciones que se ejecutan en clústeres de Kubernetes. Utiliza análisis de imágenes de contenedores y análisis de código para identificar vulnerabilidades y configuraciones incorrectas en las aplicaciones, asegurando que estén libres de brechas de seguridad.

Además, StackRox cuenta con capacidades avanzadas de detección de amenazas en tiempo real. Utiliza aprendizaje automático y análisis de comportamiento para identificar actividades maliciosas en tiempo real. Esto incluye la detección de ataques, anomalías en el tráfico de red y comportamientos sospechosos en los contenedores y aplicaciones en ejecución, lo que permite una respuesta rápida y eficiente ante posibles amenazas.

La herramienta también permite definir políticas de seguridad personalizadas. Esto significa que se pueden establecer reglas y políticas adaptadas a los requisitos específicos de cada organización. Estas políticas pueden abarcar desde reglas de acceso a redes hasta políticas de actualización y parches, lo que proporciona un control granular sobre la seguridad en el entorno de Kubernetes.

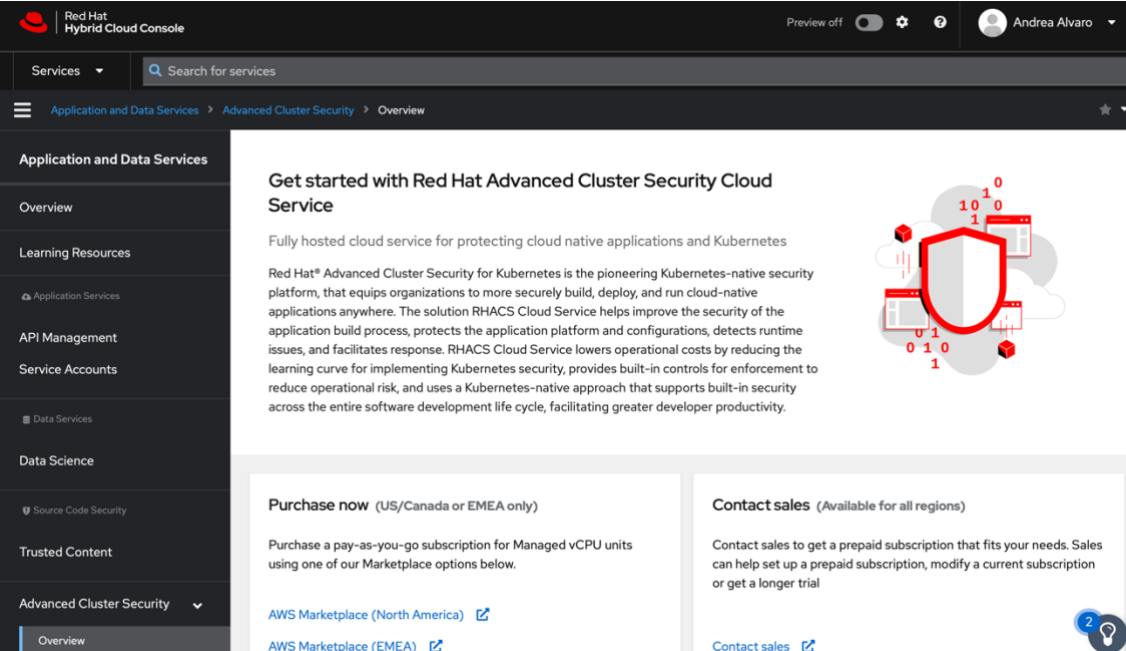
Otra característica destacable es la capacidad de evaluación de riesgos y cumplimiento normativo que ofrece StackRox. Proporciona una evaluación continua de los riesgos y el cumplimiento normativo de las aplicaciones y los clústeres de Kubernetes. Esto ayuda a identificar y remediar brechas de cumplimiento, y también proporciona informes detallados para auditorías y *reporting*.

StackRox también ofrece mitigación automática de amenazas. En caso de detectar una amenaza, la herramienta puede tomar medidas automáticas para mitigarla. Esto incluye la implementación de políticas de acceso, reglas de firewall y el bloqueo de contenedores comprometidos, basándose en reglas predefinidas o personalizadas.

Por último, StackRox se integra sin problemas con herramientas de *DevOps* existentes. Esto permite una fácil integración en los flujos de trabajo de desarrollo y despliegue. Puede integrarse con herramientas de orquestación de contenedores, como Kubernetes, y herramientas de CI/CD, facilitando la detección temprana de problemas de seguridad durante el proceso de desarrollo y despliegue.

StackRox proporciona una amplia gama de características destacables en términos de seguridad en el entorno de Kubernetes. Su enfoque en la protección de aplicaciones, la detección de amenazas en tiempo real, las políticas de seguridad personalizadas, la evaluación de riesgos y cumplimiento normativo, la mitigación automática de amenazas y la integración con herramientas de *DevOps* hacen de StackRox una solución integral para garantizar la seguridad en entornos Kubernetes.

En la ilustración 24, se observa el *dashboard* desde donde se puede configurar StackRox. Para ello es necesario tener cuenta en Red Hat y contratar el programa más adecuado a las necesidades del proyecto.



The screenshot shows the Red Hat Hybrid Cloud Console interface. The top navigation bar includes the Red Hat logo, the text 'Red Hat Hybrid Cloud Console', a 'Preview off' toggle, and a user profile for 'Andrea Alvaro'. Below the navigation bar is a search bar and a breadcrumb trail: 'Application and Data Services > Advanced Cluster Security > Overview'. The main content area is titled 'Get started with Red Hat Advanced Cluster Security Cloud Service' and describes it as a 'Fully hosted cloud service for protecting cloud native applications and Kubernetes'. It highlights that Red Hat Advanced Cluster Security for Kubernetes is a pioneering security platform that improves the security of the application build process, protects the application platform and configurations, detects runtime issues, and facilitates response. Below the text are two call-to-action boxes: 'Purchase now (US/Canada or EMEA only)' and 'Contact sales (Available for all regions)'. The 'Purchase now' box offers a pay-as-you-go subscription for Managed vCPU units using one of the Marketplace options below, with links for 'AWS Marketplace (North America)' and 'AWS Marketplace (EMEA)'. The 'Contact sales' box offers a prepaid subscription that fits your needs, with a 'Contact sales' link. A sidebar on the left contains navigation links for 'Application and Data Services', 'Overview', 'Learning Resources', 'Application Services', 'API Management', 'Service Accounts', 'Data Services', 'Data Science', 'Source Code Security', 'Trusted Content', and 'Advanced Cluster Security' (with 'Overview' selected).

Ilustración 24. Dashboard RedHat StackRox

5.8.3 Resultados

Analizando la herramienta, se descubre que StackRox es una herramienta de pago y procede a haber un periodo de prueba. Sin embargo, por petición de la empresa se analiza y se descarta para la instalación. Aun así, comentan la necesidad de realizar la comparativa a nivel de tiempo de ejecución. Este obstáculo será determinante para evaluar la viabilidad de esta herramienta en proyectos futuros. Dado que no es de código abierto, se limitará su uso a proyectos más grandes. Por lo tanto, se descarta su empleo en la demostración e instalación.

5.9 Iteración 9 – Análisis Sysdig Secure

5.9.1 Tareas US 12 y US 14

En esta iteración se realizarán las siguientes tareas de la US 12 y US 14 (Análisis Sysdig Secure, Guía de instalación):

- 01. Análisis Sysdig – funcionamiento herramienta*
- 02. Instalar Sysdig*
- 03. Redactar Instalación Sysdig Secure — Guía de instalación*
- 03. Configurar Sysdig*
- 04. Buenas prácticas Sysdig Secure*
- 05. Redactar Buenas prácticas Sysdig Secure*

5.9.2 Desarrollo

Durante esta iteración, se realizó el análisis de la herramienta Sysdig Secure, se observa que no dispone de versión gratuito. Por tanto, de igual forma que para StackRox se modifican las tareas en la iteración 8 US8 y analizan las características más destacables para poder realizar las comparativas además de ver videos explicativos del funcionamiento e instalación para proporcionar la información necesaria a la hora de comparar.

Sysdig Secure

Sysdig Secure [42] es una solución de seguridad diseñada específicamente para entornos de contenedores y orquestadores de contenedores, como Kubernetes. Es desarrollada por la empresa *Sysdig*, que se especializa en la monitorización y seguridad de entornos de contenedores y microservicios. La herramienta no ofrece versión gratuita.

Sysdig Secure proporciona una amplia gama de capacidades de seguridad para garantizar la protección de tus aplicaciones y entornos en contenedores. Algunas de las características destacadas de Sysdig Secure incluyen:

- **Visibilidad completa:** Ofrece una visibilidad exhaustiva de tus entornos de contenedores y microservicios, permitiendo monitorear y analizar en tiempo real la actividad de tus aplicaciones, el tráfico de red y los eventos del sistema.
- **Detección y prevención de amenazas en tiempo real:** Utiliza técnicas avanzadas de detección de amenazas para identificar y responder a ataques y comportamientos maliciosos en tiempo real. Esto incluye la detección de actividades sospechosas, comportamientos anómalos y ataques cibernéticos en los entornos de contenedores.
- **Escaneo de vulnerabilidades:** Realiza escaneos automáticos de imágenes de contenedores y análisis de vulnerabilidades para identificar brechas de seguridad y configuraciones incorrectas en las aplicaciones en contenedores. Esto ayuda a garantizar que las imágenes de contenedores sean seguras y libres de vulnerabilidades conocidas.
- **Cumplimiento normativo:** Proporciona características de cumplimiento normativo para asegurar que las aplicaciones cumplan con los estándares y regulaciones de seguridad requeridos. Sysdig Secure ayuda a auditar y monitorear el cumplimiento de políticas y reglas específicas.
- **Análisis forense y respuesta a incidentes:** Facilita el análisis forense y la respuesta a incidentes de seguridad al proporcionar registros detallados y la capacidad de investigar eventos pasados. Esto permite realizar análisis retrospectivos y obtener información valiosa para la respuesta y la resolución de problemas.
- **Integración con herramientas de CI/CD:** Se integra sin problemas con herramientas de integración y entrega continuas (CI/CD) para garantizar que los procesos de desarrollo y despliegue se realicen de manera segura. Sysdig Secure puede ser incorporado a los flujos de trabajo de *DevOps* existentes para proporcionar seguridad desde la etapa de desarrollo hasta el despliegue.

Sysdig Secure es una solución de seguridad completa y portante para entornos de contenedores y microservicios. Proporciona visibilidad, detección de amenazas, escaneo de vulnerabilidades, cumplimiento normativo, análisis forense y respuesta a incidentes, y se integra con herramientas de CI/CD. Todo esto ayuda a proteger las aplicaciones y entornos en contenedores de manera eficiente y segura.

5.9.3 Resultados

Analizando la herramienta, se descubre que Sysdig Secure es otra herramienta de pago poseyendo de igual forma un periodo de prueba. Sin embargo, se comenta con la empresa y proceden a las mismas conclusiones que en la iteración anterior. Por tanto, en cuanto a la herramienta se procede a compararlas de igual forma que StackRox para realizar la tabla

comparativa a nivel de tiempo de ejecución. La elección de esta herramienta dependerá de las necesidades de la empresa y del cliente al igual que del presupuesto que disponen. Debido al elevado precio se seleccionará la herramienta para grandes proyectos o por necesidad. En este caso se descarta la posibilidad de añadir a la guía de instalación su instalación y la demostración.

5.10 Iteración 10 – Análisis de las herramientas: Falco, StackRox, Sysdig

5.10.1 Tareas US 13

En esta iteración se realizarán las siguientes tareas de la US 13 (Comparativa herramientas Falco, Sysdig Secure, StackRox):

01. Analizar y seleccionar características interesantes para comparar las herramientas
02. Crear tabla comparativa de las herramientas

5.10.2 Desarrollo

Se analizan las características más pertinentes para realizar la comparativa más acertada de las herramientas. Además de sugerir los casos de usos más adecuados. Por otro lado, se destacarán las buenas prácticas únicamente de Falco debido a los contratiempos para así conseguir mejorar su uso y eficacia a los futuros proyectos.

5.10.3 Resultados

Como resultado de este proyecto hemos realizado los siguientes productos:

- **Tabla comparativa de las herramientas Falco, StackRox y Sysdig Secure**

Característica	Falco	StackRox	Sysdig Secure
Facilidad de instalación	+++	++	++
Facilidad de Uso	++	++	+
Características Destacables	Detección de comportamientos y actividades maliciosas en tiempo real.	Escaneo de vulnerabilidades en imágenes de contenedores. Integración con herramientas de <i>DevOps</i> .	Visibilidad completa de entornos de contenedores y microservicios. Detección y prevención de amenazas en tiempo real.
Open Source	Sí	No	No
Facilidad de Instalación	Moderada	Moderada	Fácil
Nivel de Conocimientos	Intermedio	Intermedio	Básico
Integración con CI/CD	Sí	Sí	Sí

Escaneo de Vulnerabilidades	No	Sí	Sí
Detección de Amenazas en Tiempo Real	Sí	Sí	Sí
Monitoreo y Visibilidad	No	Sí	Sí
Análisis Forense y Respuesta a Incidentes	No	No	Sí
Cumplimiento Normativo	No	Sí	Sí
Soporte Técnico	Comunidad	Soporte empresarial	Soporte empresarial
Interfaz de usuario	No	Si	Si
Integración con SIEM	Si	Si	Si
Automatización de acciones de respuesta	No ofrece	Permite automatizar acciones de respuesta	Ofrece automatización de acciones de respuesta
Escalable	Si	Si	Si
Generación de alertas en tiempo real	Si	Si	Si
Auditoria de politicas	Si	Si	Si

Tabla 8. Comparativa Falco, StackRox y Sysdig Secure

La tabla 9 que incluye los casos de uso de las herramientas Falco, StackRox y Sysdig Secure, así como la dimensión de los tipos de proyectos, el presupuesto necesario y la facilidad de uso:

Herramienta	Casos de Uso
Falco	Detección de comportamientos y actividades maliciosas en tiempo real en entornos de contenedores.
	Monitoreo de eventos y actividades sospechosas dentro de los contenedores y sistemas operativos subyacentes.
	Generación de alertas y notificaciones en tiempo real para responder rápidamente a posibles amenazas de seguridad.
	Cumplimiento normativo y auditoría de políticas de seguridad en entornos de contenedores.
	Integración con herramientas de orquestación y CI/CD para una detección temprana de problemas de seguridad durante el desarrollo y despliegue de aplicaciones en contenedores.

StackRox	Escaneo de vulnerabilidades en imágenes de contenedores para identificar y corregir brechas de seguridad.
	Evaluación continua de riesgos y cumplimiento normativo en entornos de Kubernetes.
	Detección y respuesta automática a amenazas de seguridad en tiempo real utilizando aprendizaje automático y análisis de comportamiento.
	Protección de aplicaciones en entornos de Kubernetes mediante el análisis de código y la gestión de vulnerabilidades.
	Integración con herramientas de <i>DevOps</i> para una implementación y administración sin problemas.
Sysdig Secure	Visibilidad completa de entornos de contenedores y microservicios para monitorear y analizar en tiempo real la actividad de las aplicaciones, el tráfico de red y los eventos del sistema.
	Detección y prevención de amenazas en tiempo real mediante técnicas avanzadas de detección de amenazas y análisis de comportamiento.
	Escaneo de vulnerabilidades en imágenes de contenedores para identificar y corregir problemas de seguridad conocidos.
	Cumplimiento normativo y auditoría para garantizar que las aplicaciones cumplan con los estándares y regulaciones de seguridad requeridos.
	Análisis forense y respuesta a incidentes para investigar y resolver problemas de seguridad pasados y actuales.
	Integración con herramientas de CI/CD para garantizar la seguridad en el proceso de desarrollo y despliegue de aplicaciones en contenedores.

Tabla 9. Casos de uso de las herramientas Falco, StackRox y Sysdig Secure

Herramientas	Tipos de Proyectos	Presupuesto Necesario	Facilidad de Uso
Falco	Todo tipo de proyectos en contenedores.	Bajo	Moderada
StackRox	Proyectos en entornos de Kubernetes.	Medio	Moderada
Sysdig Secure	Todo tipo de proyectos en contenedores y microservicios.	Alto	Fácil

Tabla 10. Casos de uso dependiendo del proyecto: Falco, StackRox y Sysdig Secure

La tabla 10 proporciona una referencia inicial para comprender la relación entre la facilidad de uso y el presupuesto necesario en diferentes herramientas o soluciones, pero se recomienda utilizarla como punto de partida y adaptarla a las necesidades y circunstancias particulares de cada proyecto.

Es importante tener en cuenta que la facilidad de uso y el presupuesto necesario pueden variar dependiendo de la experiencia y los conocimientos del usuario, así como de los requisitos y la infraestructura específica de cada proyecto. Para proporcionar una estimación general de estos aspectos, se presenta la tabla 10, donde se clasifica la facilidad de uso y el presupuesto necesario para diferentes herramientas o soluciones.

Es necesario considerar que esta clasificación es una estimación general y puede variar en función de diversos factores. La facilidad de uso puede depender de la familiaridad del usuario con la tecnología, la complejidad de la herramienta y la disponibilidad de recursos de capacitación. Por otro lado, el presupuesto necesario puede verse afectado por el costo de adquisición o implementación de la herramienta, los gastos de mantenimiento y soporte, así como los requerimientos de infraestructura y licencias.

Es recomendable revisar detenidamente cada herramienta o solución, evaluar en profundidad las características y considerar los requisitos y limitaciones específicos del proyecto antes de tomar una decisión. Además, es importante contar con realizar pruebas o evaluaciones en el entorno adecuado antes de realizar una implementación a gran escala.

5.11 Iteración 11 – Redacción guía de instalación

5.11.1 Tareas US 14

En esta iteración se realizarán las siguientes tareas de la US 14 (Guía de instalación):

01. Guía de instalación

5.11.2 Desarrollo

Durante esta iteración, se realiza la redacción de la guía de instalación con el objetivo de proporcionar a los usuarios una mayor comprensión de las herramientas, así como facilitar la instalación eficiente de las mismas al tener toda la información recopilada en un solo lugar. El manual permitirá agilidad en el proceso y maximizará el aprovechamiento de las herramientas al configurarlas según las necesidades específicas.

5.11.3 Resultado

- **Guía de Instalación de las herramientas y buenas prácticas recomendadas.** Se encuentra en el *anexo 5*.

5.12 Iteración 12 – Preparación para de demostración

5.12.1 Tareas US 15

En esta iteración se realizarán las siguientes tareas de la US 15 (Preparación demostración):

01. Selección herramientas para la demostración
02. Preparar demostración con las herramientas
03. Guion para la demostración

5.12.2 Desarrollo

Durante esta iteración, se han seleccionado las herramientas necesarias para la demostración. Dado que el proyecto ha experimentado cambios, se han elegido las herramientas más adecuadas para su uso. En esta ocasión, se ha decidido utilizar las siguientes herramientas:

- Kube-bench
- Kube-hunter
- Trivy
- Falco

La implementación de estas herramientas permite comprender la importancia y la necesidad de proteger Kubernetes en diferentes niveles.

5.12.3 Resultados

Se obtiene un guion para preparar la demostración de las herramientas.

6. Estudio económico

6.1 Desglose de costes

6.1.1 Costes materiales

En cuanto a los gastos del proyecto, se pueden identificar distintas secciones según los materiales que se han utilizado durante el desarrollo, siendo distribuidos de la siguiente manera, en la tabla 11:

	Precio	Vida Útil	Tiempo usado	Coste Real
MacBook Pro 13"	2240€	5 años [43]	5 meses	187€
Periféricos	100€	6 años [44]	5 meses	7€
Pantalla 27"	120€	30 000 horas [45]	335h	1,34€
			Total	195,34€

Tabla 11. Costes materiales

6.1.2 Costes humanos

En la tabla 12, se presentan los costes relacionados con el personal requerido para llevar a cabo el proyecto, los cuales están distribuidos según los roles desempeñados y el tiempo dedicado en cada etapa del proceso de desarrollo.

	Coste/hora	Tiempo Usado	Coste Total
Project manager	20€/h	25	440€
Desarrollador/Analista	18€/h	310	5580€
	Total	335 h	6.020€

Tabla 12. Costes humanos

6.1.3 Costes de infraestructura

A lo largo de los 5 meses de desarrollo del proyecto, se requiere un entorno de trabajo adecuado, lo cual conlleva costos de infraestructura que se detallan en la tabla 13:

	Precio/mes	Tiempo de uso	Coste
Alquiler	111€/mes	5 meses	555€
Agua	7,6€/mes *	5 meses	38€
Luz	55€/mes **	5 meses	275€
Internet	30€/mes **	5 meses	150€
	Total	335 h	1.018€

Tabla 13. Costes de infraestructura

* Se estiman valores medios. Los precios se basan para una persona al mes

** Se estima mayor contratación de luz e internet puesto que se gasta mayoritariamente para realizar el proyecto.

6.1.4 Costes totales

Finalmente, calculamos y recopilamos la suma total de los gastos asociados al desarrollo del proyecto en la tabla 14.

Costes Material	Costes Humanos	Coste de Infraestructura	Total
195,34€	6.020€	1.018€	7.233,34€

Tabla 14. Costes totales

7. Resultados

En primer lugar, al finalizar el proyecto se han obtenido los siguientes productos:

- **Guía de Instalación de las herramientas y buenas prácticas recomendadas**
- **Tabla comparativa de las herramientas Falco, StackRox y Sysdig Secure**
- **Tablas comparativas de herramientas a distintos niveles**
- **Repositorio en GitHub con todos los elementos**

Las tablas comparativas permiten seleccionar las herramientas adecuadas dependiendo del proyecto y de sus necesidades. Además, la guía de instalación permite mayor eficacia para el equipo a la hora de instalar dichas herramientas.

Se ha creado un repositorio en GitHub que alberga todos los ejemplos utilizados a lo largo del proyecto, así como las memorias y la guía de instalación destinada a los usuarios. El repositorio está disponible en el siguiente enlace: <https://github.com/andreaalvaro/SeguridadK8s.git>

Las fases de análisis previas a la implementación desempeñaron un papel crucial en el desarrollo exitoso de este proyecto al proporcionar una base sólida para realizar comparativas coherentes y consistentes. La implementación se llevó a cabo sin contratiempos significativos, aunque se realizaron algunos ajustes en la planificación inicial. A lo largo del proceso, surgieron obstáculos que requirieron intervención y una redefinición de los objetivos, especialmente en lo que respecta al uso de ciertas herramientas debido a su coste. Al reajustar los objetivos, se eliminaron riesgos en el proyecto en cuanto a los plazos de prueba de las herramientas, ya que se modificó la selección de la instalación de las herramientas y de la demo de Sysdig Secure y StackRox.

En cuanto al resultado de la planificación del proyecto, se pueden observar las modificaciones que afectaron la planificación inicial y cómo se reflejaron en el resultado final en las ilustraciones siguientes. En la ilustración 25, se muestra un gráfico que representa la comparativa entre el tiempo estimado y el tiempo real en función de las horas dedicadas al proyecto. Este gráfico proporciona una visualización rápida de cómo ha evolucionado el proyecto en términos de cumplimiento de los plazos previstos.

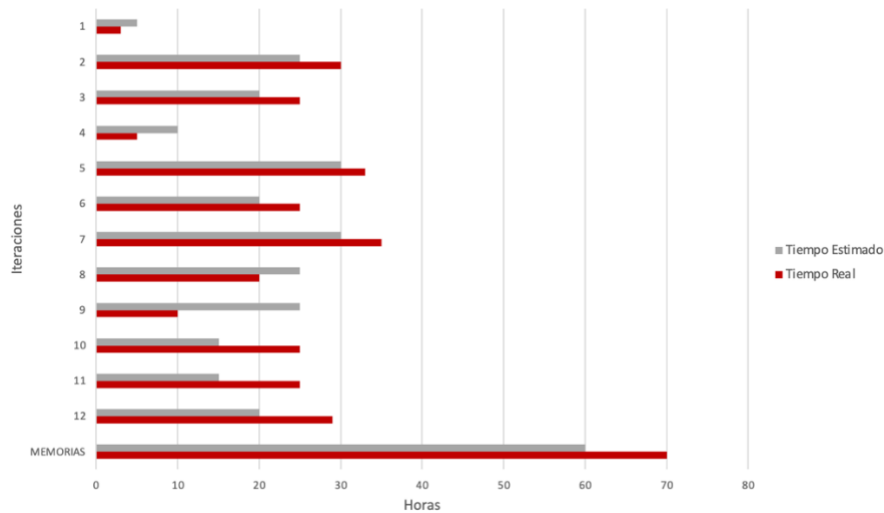


Ilustración 25. Planificación inicial y final

Las iteraciones que han experimentado mayores desviaciones en este proyecto son la 8 y 9. Estas iteraciones se vieron afectadas principalmente debido a que se utilizó una herramienta de pago, lo cual implicó descartar la instalación de dichas herramientas y por consiguiente reducir las tareas de las historias de usuario correspondientes a esas iteraciones. Además, las iteraciones 10, 11 y 12 también sufrieron algunas desviaciones, ya que no se tuvo en cuenta suficiente margen para posibles errores.

Finalmente, en la ilustración 26 se presenta el tiempo total invertido en este proyecto.

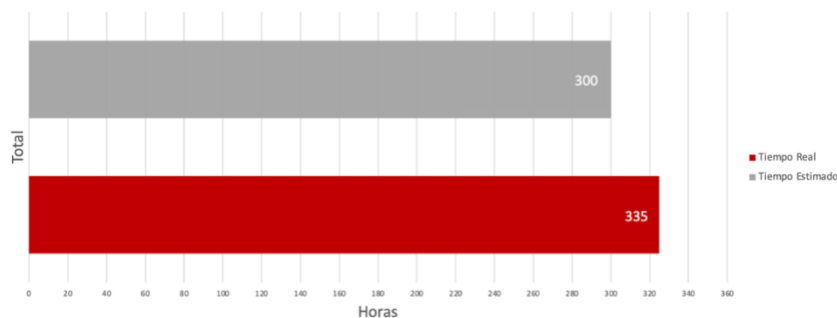


Ilustración 26. Planificación total

En este proyecto, se ha llevado a cabo la implementación de 12 iteraciones, las cuales se dividieron en 41 tareas y se realizaron las correspondientes pruebas de aceptación. Por último, se plantearon acciones correctivas para mejorar y añadir características adicionales a las tablas comparativas.

A lo largo del proyecto se realizan diversas pruebas para asegurar el correcto funcionamiento de las herramientas instaladas. Estas pruebas se dividen en pruebas funcionales y pruebas de la guía de instalación, que garantizan tanto la operatividad de las herramientas como la eficacia de los pasos de instalación. Algunos ejemplos de las pruebas funcionales incluyen:

- **Pruebas de integración:** Se verifica la interacción y compatibilidad entre las herramientas instaladas y los componentes del entorno, como Kubernetes. Se asegura que las herramientas se integren adecuadamente y funcionen de manera coherente dentro del entorno de Minikube. Por ejemplo, se comprueba si se crean correctamente los pods y si las herramientas se instalan sin problemas.
- **Pruebas de funcionalidad:** Se evalúa el comportamiento y las funcionalidades de las herramientas. Se ejecutan escenarios específicos para comprobar que las herramientas realicen las acciones esperadas y proporcionen los resultados correctos. Por ejemplo, se pueden realizar pruebas para verificar la detección de vulnerabilidades por parte de Trivy utilizando imágenes de nginx de diferentes versiones, la generación y gestión de secretos con Sealed Secrets y Vault, o el monitoreo de actividades sospechosas mediante la creación de alertas con Falco. Asimismo, con Kube-Bench se instalan y escanean los pods creados, y se verifica que se obtengan los resultados de forma correcta.

Estas pruebas funcionales permiten verificar que las herramientas cumplen con sus funcionalidades específicas y que operan adecuadamente dentro del entorno de Minikube. Además, se realizan pruebas adicionales, como el proceso de reinstalación, para asegurarse de que las herramientas sigan funcionando correctamente incluso después de un nuevo despliegue.

Antes de publicar la guía de instalación, se realiza una validación exhaustiva y pruebas en un entorno de prueba. Se siguen los pasos descritos en la guía y se verifican los resultados obtenidos. Durante esta fase, se identifican posibles errores, omisiones o dificultades en el proceso, los cuales son corregidos antes de que los usuarios finales utilicen la guía. La guía se somete a una verificación completa para garantizar su completitud y se realiza una instalación desde cero siguiendo los pasos proporcionados. Este proceso permite confirmar la precisión de la guía y la claridad y concisión de los procedimientos descritos.

Una vez finalizada la guía, se entrega a NTTDATA el conjunto de herramientas para que su equipo realice pruebas exhaustivas en su propio entorno. Se solicita al equipo de NTTDATA que evalúe la rapidez de instalación y reporte los errores más comunes encontrados durante el proceso. Estos informes de errores se utilizan para mejorar la documentación y proporcionar soluciones a los problemas recurrentes, mejorando así la experiencia de instalación.

Este proceso integral de pruebas y revisión garantiza que las herramientas estén configuradas correctamente y listas para su uso en el entorno específico de NTTDATA. Además, ayuda a identificar posibles problemas o errores comunes, asegurando una instalación exitosa y un funcionamiento óptimo de las herramientas en el entorno real.

Se observa una mayor eficacia y rapidez en el proceso de instalación, ya que todas las soluciones están incluidas en un único documento y no es necesario realizar búsquedas individuales o investigaciones adicionales. Sin embargo, es importante trabajar en la mejora continua de la guía de instalación para abordar y proporcionar soluciones a los fallos identificados durante el proceso. Es fundamental evaluar la eficacia de la guía de instalación mediante un seguimiento y gestión de problemas.

Durante las pruebas, se registra y se realiza un seguimiento de cualquier problema o error identificado. Estos problemas se documentan y se resuelven para garantizar que la guía de instalación funcione correctamente y, en caso necesario, se mejore. El proceso de seguimiento y gestión de problemas nos permite ser más eficaces al proporcionar soluciones en la guía y mejorar la experiencia de instalación para todos los usuarios. Al registrar y abordar los problemas identificados, podemos identificar áreas de mejora y realizar ajustes para garantizar que la guía sea clara, precisa y efectiva.

Este enfoque de seguimiento y gestión de problemas nos permite mantener un ciclo de mejora continua, donde aprendemos de las pruebas y experiencias para perfeccionar la guía de instalación. De esta manera, podemos brindar soluciones más eficientes y mejorar la experiencia de instalación para todos los usuarios, asegurando que puedan implementar las herramientas de manera exitosa y sin dificultades innecesarias.

8. Conclusiones

En primer lugar, se logró satisfactoriamente alcanzar los objetivos establecidos para este proyecto, que incluyeron realizar una comparativa de herramientas en términos de tiempo de ejecución, escaneo de imágenes, cumplimiento normativo, gestión de secretos y políticas. Además, se creó una guía de instalación detallada para dichas herramientas.

Tras analizar los resultados obtenidos en este proyecto, se concluye que la guía de instalación puede ser utilizada en entornos empresariales reales para mejorar la configuración e instalación de seguridad en Kubernetes, al mismo tiempo que reduce el tiempo de investigación para los miembros del equipo.

Por último, este proyecto ha sido validado por expertos en seguridad de Kubernetes y profesionales de NTTDATA, quienes confirmaron que las herramientas propuestas son adecuadas y válidas para proteger Kubernetes. Esto significa que las comparativas realizadas son completas y pueden aplicarse en entornos reales para resolver problemas reales en la protección satisfactoria del entorno de Kubernetes.

8.1 Propuesta de mejora

Aunque tiene un gran alcance de herramientas este proyecto todavía se pueden tomar varias acciones para mejorarlo y convertirlo en un producto más completo:

- **US 16. Ampliar la cobertura de herramientas de seguridad analizadas.** El proyecto se centra en una lista específica de herramientas, por ello es interesante ampliar dicha lista para incluir más opciones y proporcionar una comparación más exhaustiva.
- **US 17. Profundizar en más niveles de seguridad.** No solo es importante analizar la seguridad en el nivel de ejecución en tiempo de ejecución (runtime), sino que también es necesario profundizar en los demás niveles, como la seguridad a nivel de código, a nivel de clúster y a nivel de infraestructura en la nube. Un aspecto relevante sería seleccionar la mejor estructura en la nube para la empresa, considerando sus necesidades y requisitos específicos.

Además, es importante ahondar en más características y funcionalidades específicas de cada herramienta de seguridad para proporcionar una descripción más detallada y un análisis más profundo. Esto permitirá evaluar de manera más precisa cuál es la opción más adecuada para el entorno de Kubernetes en cuestión, teniendo en cuenta aspectos

como la detección de amenazas, la gestión de políticas de seguridad, la respuesta automatizada y otras funcionalidades relevantes.

Al profundizar en estos aspectos, se logrará una comprensión más completa de las herramientas de seguridad y se podrá tomar una decisión informada sobre cuál es la mejor opción para garantizar la protección y la integridad del entorno de Kubernetes.

- **US 18. Añadir en la guía de instalación,** la instalación de todas las herramientas que aparecen en la ilustración 5. Y así proteger Kubernetes a todos los niveles.
- **US 19.** Incluir **recomendaciones** de todas las herramientas de seguridad mencionadas.
- **US 20. Analizar la integración con otras herramientas de seguridad.** Considerar analizar la integración las herramientas de seguridad con otras soluciones de seguridad, como herramientas de monitoreo y análisis de registros, para proporcionar una evaluación más completa del panorama de seguridad de Kubernetes.
- **US 21. En un entorno empresarial, se propone llevar a cabo pruebas de la guía de instalación para evaluar la eficacia y facilidad de uso** de las herramientas de seguridad de Kubernetes. Para ello, se sugiere dividir al equipo especializado en seguridad de Kubernetes en dos grupos: Grupo 1 y Grupo 2.

En el Grupo 1, los miembros instalarán las herramientas sin utilizar la guía de usuario, mientras que en el Grupo 2, se permitirá el uso del manual durante la instalación. Se compararán y analizarán los tiempos de instalación de ambos grupos para determinar si la utilización de la guía de instalación mejora la eficacia y la facilidad de uso.

Esta metodología permitirá obtener datos cuantitativos y cualitativos sobre la influencia del manual en el proceso de instalación de las herramientas de seguridad de Kubernetes. De esta manera, se podrán identificar posibles brechas en la documentación o áreas de mejora para garantizar una implementación más eficiente y efectiva de las herramientas de seguridad.

- **US 22. Pruebas de seguridad.** Además, es posible realizar pruebas de seguridad para evaluar la capacidad de las herramientas en la detección y respuesta a amenazas de seguridad. Estas pruebas pueden incluir pruebas de penetración, detección de intrusiones y análisis de vulnerabilidades, con el objetivo de evaluar la efectividad de las herramientas en la protección del entorno.

9. Bibliografía

- [1] "Ciberamenaza Mapa Tiempo Real ES", Kaspersky. [En línea]. Available: <https://cybermap.kaspersky.com/es> [Ultimo acceso: Junio 2023]
- [2] Harris, Chandler. "Comparación entre la arquitectura monolítica y la arquitectura de microservicios", Atlassian. [En línea]. Available: <https://www.atlassian.com/es/microservices/microservices-architecture/microservices-vs-monolith> [Ultimo acceso: mayo 2023]
- [3] Singh, Harender. Casanova, Alfonso. Elvira, Jesús. Garrido, Pedro. Soto, Óscar. "Arquitectura Spotify (Microservicios) – Grupo 0", aprendearquitecturasoftware [En línea]. Available: <https://aprendearquitecturasoftware.wordpress.com/2018/09/27/arquitectura-spotify-microservicios/> [Ultimo acceso: septiembre 2018]
- [4] Cuemby, "Grandes Empresas que usan Kubernetes", Medium. [En línea]. Available: <https://medium.com/@cuemby/grandes-empresas-que-usan-kubernetes-d73d56334243> [Ultimo acceso: octubre 2019]
- [5] AWS, "¿Qué son los microservicios?" [En línea]. Available: <https://aws.amazon.com/es/microservices/> [Ultimo acceso: mayo 2023]
- [6] Buchanan, Ian. "Comparación de contenedores y máquinas virtuales", Atlassian. [En línea]. Available: <https://www.atlassian.com/es/microservices/cloud-computing/containers-vs-vms> [Ultimo acceso: mayo 2023]
- [7] Docker. [En línea]. Available: <https://www.docker.com> [Ultimo acceso: mayo 2023]
- [8] Cloud Native Computing Foundation "Orquestación de contenedores para producción: introducción", Kubernetes. [En línea]. Available: <https://kubernetes.io/es/> [Ultimo acceso: 2023]
- [9] "CNCF SURVEY 2020, Use of containers in production has increased by 300% since 2016" [En línea]. Available: https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf [Ultimo acceso: 2020]
- [10] "9 Insights on Real-World Container User", DataDog [En línea]. Available: <https://www.datadoghq.com/container-report/> [Ultimo acceso: 2022]
- [11] "2021 Global Tech Outlook: un informe de Red Hat", RedHat. [En línea]. Available: <https://www.redhat.com/es/resources/2021-global-tech-outlook-detail> [Ultimo acceso: febrero 2021]
- [12] "Companies using Kubernetes", Enlyft. [En línea]. Available: <https://enlyft.com/tech/products/kubernetes> [Ultimo acceso: 2023]
- [13] "El concepto de DevOps", Red Hat. [En línea]. Available: <https://www.redhat.com/es/topics/devops#:~:text=DevOps%20es%20un%20modo%20de,de%20los%20servicios%20de%20TI> [Ultimo acceso: mayo 2022]
- [14] "¿Qué es DevSecOps?", VMWare. [En línea]. Available: [https://www.vmware.com/es/topics/glossary/content/devsecops.html#:~:text=«DevSecOps»%20\(a%20breviatura%20de%20«diseñar%20aplicaciones%20sólidas%20y%20seguras](https://www.vmware.com/es/topics/glossary/content/devsecops.html#:~:text=«DevSecOps»%20(a%20breviatura%20de%20«diseñar%20aplicaciones%20sólidas%20y%20seguras). [Ultimo acceso: 2023]
- [15] Marusiak, Warren. "Como aplicar DevOps", Atlassian. [En línea]. Available: <https://www.atlassian.com/es/devops/what-is-devops/how-to-start-devops>

-
- [16] Raeburn, Alicia. "La programación extrema (XP) produce resultados, pero ¿es la metodología adecuada para ti?", Asana. [En línea]. Available: <https://asana.com/es/resources/extreme-programming-xp> [Ultimo acceso: noviembre 2022]
- [17] Atlassian Trello, [En línea]. Available: <https://trello.com> [Ultimo acceso: 2023]
- [18] "Microsoft Excel", Microsoft 365. [En línea]. Available: <https://www.microsoft.com/es-es/microsoft-365/excel> [Ultimo acceso: 2023]
- [19] "State of Kubernetes security report", Red Hat. [En línea o PDF]. Available: <https://www.redhat.com/rhdc/managed-files/cl-state-kubernetes-security-report-262667-202304-en.pdf> [Ultimo acceso: 2023]
- [20] "Overview", Kubernetes. [En línea]. Available: <https://kubernetes.io/docs/concepts/overview/> [Ultimo acceso: enero 2023]
- [21] "Kubernetes Components", Kubernetes. [En línea]. Available: <https://kubernetes.io/docs/concepts/overview/components/> [Ultimo acceso: mayo 2019]
- [22] "Overview of Cloud Native Security", Kubernetes. [En línea]. Available: <https://kubernetes.io/docs/concepts/security/overview/> [Ultimo acceso: septiembre 2022]
- [23] "Seguridad en la nube de AWS", Amazon Web Services. [En línea]. Available: <https://aws.amazon.com/es/security/> [Ultimo acceso: 2023]
- [24] INCIBE, ¿Por qué adoptar un enfoque proactivo en nuestra organización? [En línea]. Available: <https://www.incibe.es/empresas/blog/adoptar-enfoque-proactivo-nuestra-organizacion> [Ultimo acceso: marzo 2023]
- [25] [En línea]. Available: <https://www.paloaltonetworks.com/blog/2018/11/defina-una-superficie-de-proteccion-para-reducir-drasticamente-la-superficie-de-ataque/?lang=es>
- [26] NGINX, [En línea]. Available: <https://www.nginx.com> [Ultimo acceso: junio 2023]
- [27] "aquasec/kube-bench", Docker. [En línea]. Available: <https://hub.docker.com/r/aquasec/kube-bench> [Ultimo acceso: junio 2023]
- [28] itaysk. "aquasecurity/ Kube-hunter", GitHub. [En línea]. Available: <https://github.com/aquasecurity/kube-hunter> [Ultimo acceso: septiembre 2022]
- [29] "Kubernetes", Vault. [En línea]. Available: <https://developer.hashicorp.com/vault/docs/platform/k8s#> [Ultimo acceso: 2023]
- [30] "bitnami-labs/ Sealed secrets. Sealed Secrets for Kubernetes", GitHub. [En línea]. Available: <https://github.com/bitnami-labs/sealed-secrets> [Ultimo acceso: june 2023]
- [31] "Policy-based control for cloud native environments", Open Policy Agent. [En línea]. Available: <https://www.openpolicyagent.org> [Ultimo acceso: 2023]
- [32] "Kyverno, Kubernetes Native Policy Management", Kyverno. [En línea]. Available: <https://kyverno.io> [Ultimo acceso: 2023]
- [33] "Docker registry", Docker. [En línea]. Available: <https://docs.docker.com/registry/> [Ultimo acceso: 2023]
- [34] "What is Harbor?", Harbor. [En línea]. Available: <https://goharbor.io> [Ultimo acceso: 2023]
- [35] "codenotary/Kube-notary", Github.

-
- [En línea]. Available: <https://github.com/codenotary/kube-notary> [Ultimo acceso: abril2023]
- [36] Lawrence, David. "What is Notary and why is it important to CNCF?", Docker [En línea]. Available: <https://www.docker.com/blog/notary-important-cncf/> [Ultimo acceso: octubre 2017]
- [37] Trivy. [En línea]. Available: <https://trivy.dev> [Ultimo acceso: 2023]
- [38] "Kubernetes", Trivy. [En línea]. Available: <https://aquasecurity.github.io/trivy/v0.28.1/docs/kubernetes/cli/scanning/> [Ultimo acceso: 2023]
- [39] "quay/clair", GitHub. [En línea]. Available: <https://github.com/quay/clair> [Ultimo acceso: june 2023]
- [40] Falco. [En línea]. Available: <https://falco.org> [Ultimo acceso: 2023]
- [41] "StackRox Documentation", StackRox RedHat. [En línea]. Available: <https://www.stackrox.io/docs/> [Ultimo acceso: 2023]
- [42] "Sysdig Secure", Sysdig. [En línea]. Available: <https://sysdig.com/products/secure/> [Ultimo acceso: 2023]
- [43] "La vida útil de la tecnología Apple puede llegar a los 12 años", ticpymes. [En línea]. Available: <https://www.ticpymes.es/tecnologia/noticias/1111246049504/vida-util-de-tecnologia-apple-puede-llegar-12-anos.1.html#> [Ultimo acceso: junio 2023]
- [44] Versus, Apple Magic Mouse 2 : análisis, características y precio. [En línea]. Available <https://versus.com/es/apple-magic-mouse-2/battery-life-hours> [Ultimo acceso: mayo 2023]
- [45] Diego de Usera, Juan. "¿Cuál es la vida útil de un monitor antes de que se rompa?", HZ hardzone [En línea]. Available: <https://hardzone.es/2019/01/13/vida-util-monitor-antes-rompa/> [Ultimo acceso: enero 2019]
- [46] "¿Cuánto se paga al mes por los suministros básicos en España?", Alegria Real State.[En línea]. Available: <https://alegria-realestate.com/es/articles/life-in-spain/how-much-is-paid-per-month-for-basic-supplies-in-spain> [Ultimo acceso: noviembre 2022]
- [47] Kyverno, "Add Network Policy" [En línea]. Available: <https://kyverno.io/policies/best-practices/add-network-policy/add-network-policy/> [Ultimo acceso: 2023]
- [48] Kyverno, "Add Labels". [En línea]. Available: <https://kyverno.io/policies/other/add-labels/add-labels/> [Ultimo acceso: 2023]
- [49] Kyverno, "Allowed Annotations" [En línea]. Available: <https://kyverno.io/policies/other/allowed-annotations/allowed-annotations/> [Ultimo acceso: 2023]
- [50] Kyverno, "Require Image Vulnerability Scans" [En línea]. Available: <https://kyverno.io/policies/other/require-vulnerability-scan/require-vulnerability-scan/>[Ultimo acceso: 2023]
- [51] Open Policy Agent, "Kubernetes Admission control", [En línea]. Available: <https://www.openpolicyagent.org/docs/v0.12.2/kubernetes-admission-control/> [Ultimo acceso: 2023]

Anexo 1: Propuesta

Nombre alumno: Andrea Álvaro

Titulación: Ingeniería Informática

Curso académico: 2022 - 2023

1. TÍTULO DEL PROYECTO

Seguridad en K8s

2. DESCRIPCIÓN Y JUSTIFICACIÓN DEL TEMA A TRATAR

Los requerimientos de seguridad están aumentando mucho en el último año, y las herramientas habituales no sirven para securizar plataformas basadas en K8s.

Se propone el estudio **Sysdig Secure**, **Falco** y **StackRox** como herramientas de seguridad para plataformas basadas en K8s.

Sysdig Secure utiliza una plataforma unificada para ofrecer seguridad, monitorización y análisis forense sobre plataformas K8s.

Falco es el primer proyecto de seguridad en tiempo de ejecución que se une al CNCF como proyecto de nivel de incubación. Falco detecta comportamientos inesperados, intrusiones y robos de datos en tiempo real

StackRox adquirida recientemente por redhat, incluye funciones de seguridad nativas de Kubernetes y elementos de DevOps.

3. OBJETIVOS DEL PROYECTO

- Configurar y desplegar las herramientas de seguridad descritas.
- Realizar una comparativa de las herramientas en base a las funcionalidades que ofrecen.
- Crear documentación que permita reproducir lo realizado y configurar las principales buenas prácticas de seguridad sobre las herramientas.

4. METODOLOGÍA

La metodología se establecerá en las primeras fases del proyecto

5. PLANIFICACIÓN DE TAREAS

Las tareas quedan predefinidas de manera global en los objetivos. Serán fijadas de forma concreta durante el desarrollo del proyecto. De forma genérica, englobarán:

- Instalar uno o varias plataformas K8s para probar las capacidades de las herramientas de seguridad.
- Crear una matriz de características que permita decidir la herramienta más adecuada según escenarios.
- Documentar los procesos de instalación, configuración y operación de las herramientas.

6. OBSERVACIONES ADICIONALES

Se trata de un proyecto de la empresa NTT Data.

Anexo 2: Historias de Usuario

En este anexo, se detallan y presentan las historias de usuario correspondientes al proyecto.

ID: 0	Título: Preparación	
<p>Descripción:</p> <p>Es necesario analizar y exponer los objetivos y problemas relacionados con la seguridad de Kubernetes y la guía de instalación.</p>		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 5 h

ID: 01	Título: Análisis entorno Kubernetes	
<p>Descripción:</p> <p>Es necesario comprender la estructura de Kubernetes. Además, investigar y analizar las herramientas existentes en el mercado previamente a la implementación del proyecto. Y así seleccionar el más adecuado para el proyecto.</p> <p>Este proceso incluye la instalación de Kubernetes y herramientas necesarias.</p>		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 15 h

ID: 02	Título: Preparación entorno Kubernetes	
<p>Descripción:</p> <p>Es necesario realizar pruebas para verificar la correcta instalación de Kubernetes. Además de configurar <i>pods</i> y <i>namespaces</i> para observar su funcionamiento.</p> <p>Este proceso incluye la elaboración de una tabla comparativa que muestra los distintos métodos de instalación disponibles, sus ventajas y desventajas con relación a los casos de uso.</p>		
Prioridad: Alta	Riesgo: Medio	Esfuerzo: 10 h

ID: 03	Título: Entorno Cloud	
<p>Descripción:</p> <p>Es necesario analizar la seguridad en las aplicaciones <i>cloud</i>. Además de entender los niveles de seguridad que existen en <i>cloud</i> puesto que la seguridad es un término amplio.</p> <p>Este proceso incluye la creación de una tabla de buenas prácticas por áreas de seguridad existentes.</p>		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 20 h

ID: 04	Título: Herramientas Kubernetes	
<p>Descripción:</p> <p>Es necesario detectar las herramientas relacionadas con seguridad nativas de Kubernetes para analizar si son suficientes para proteger el entorno de Kubernetes. Si no es el caso, se investigan herramientas de terceros.</p>		
Prioridad: Media	Riesgo: Bajo	Esfuerzo: 8 h

ID: 05	Título: Selección herramientas	
<p>Descripción:</p> <p>Es necesario llevar a cabo la selección de las herramientas a utilizar, dado que existen numerosas opciones y resulta imposible abarcar todas ellas. El diagrama de herramientas seleccionadas constituirá una referencia visual valiosa que ayudará a garantizar que todas las herramientas necesarias estén contempladas y a mantener una visión global del proyecto en términos de tecnologías y soluciones adoptadas.</p>		
Prioridad: Media	Riesgo: Medio	Esfuerzo: 2 h

ID: 06	Título: Herramientas a nivel de clúster	
<p>Descripción:</p> <p>Es necesario investigar sobre las herramientas a nivel de clúster y proceder a la instalación de dichas herramientas.</p> <p>Este proceso incluye la configuración de las herramientas.</p>		
Prioridad: Medio	Riesgo: Alto	Esfuerzo: 25 h

ID: 07	Título: Tabla comparativa – nivel clúster	
<p>Descripción:</p> <p>Es necesario realizar una tabla comparativa de las herramientas para seleccionar la más adecuada en cada ámbito.</p>		
Prioridad: Medio	Riesgo: Bajo	Esfuerzo: 5 h

ID: 08	Título: Herramientas a nivel de tiempo de ejecución	
<p>Descripción:</p> <p>Es necesario investigar sobre las herramientas a nivel de tiempo de ejecución y proceder a la instalación de dichas herramientas. Sin embargo, el apartado de tiempo de ejecución no se analizará en esta iteración. Se realizarán en las siguientes iteraciones.</p>		

Este proceso incluye la configuración de las herramientas.		
Prioridad: Medio	Riesgo: Alto	Esfuerzo: 18 h

ID: 09	Título: Tabla comparativa – Tiempo de ejecución	
Descripción: Es necesario realizar una tabla comparativa de las herramientas para seleccionar la más adecuada en el ámbito de escaneo de imagen.		
Prioridad: Medio	Riesgo: Bajo	Esfuerzo: 2 h

ID: 10	Título: Análisis de la Herramienta Falco	
Descripción: Es necesario realizar un análisis en profundidad de la herramienta Falco. Entender el funcionamiento y la configuración de dicha herramienta. Este proceso incluye la instalación de Falco y su configuración.		
Prioridad: Medio	Riesgo: Alto	Esfuerzo: 20 h

ID: 11	Título: Análisis StackRox	
Descripción: Es necesario realizar un análisis en profundidad de la herramienta StackRox. Entender el funcionamiento y la configuración de dicha herramienta. Este proceso incluye si es posible la instalación de StackRox y su configuración.		
Prioridad: Alta	Riesgo: Alto	Esfuerzo: 20 h

ID: 12	Título: Análisis Sysdig Secure	
Descripción: Es necesario realizar un análisis en profundidad de la herramienta Sysdig Secure. Entender el funcionamiento y la configuración de dicha herramienta. Este proceso incluye si es posible la instalación de Sysdig Secure y su configuración.		
Prioridad: Alta	Riesgo: Alto	Esfuerzo: 20 h

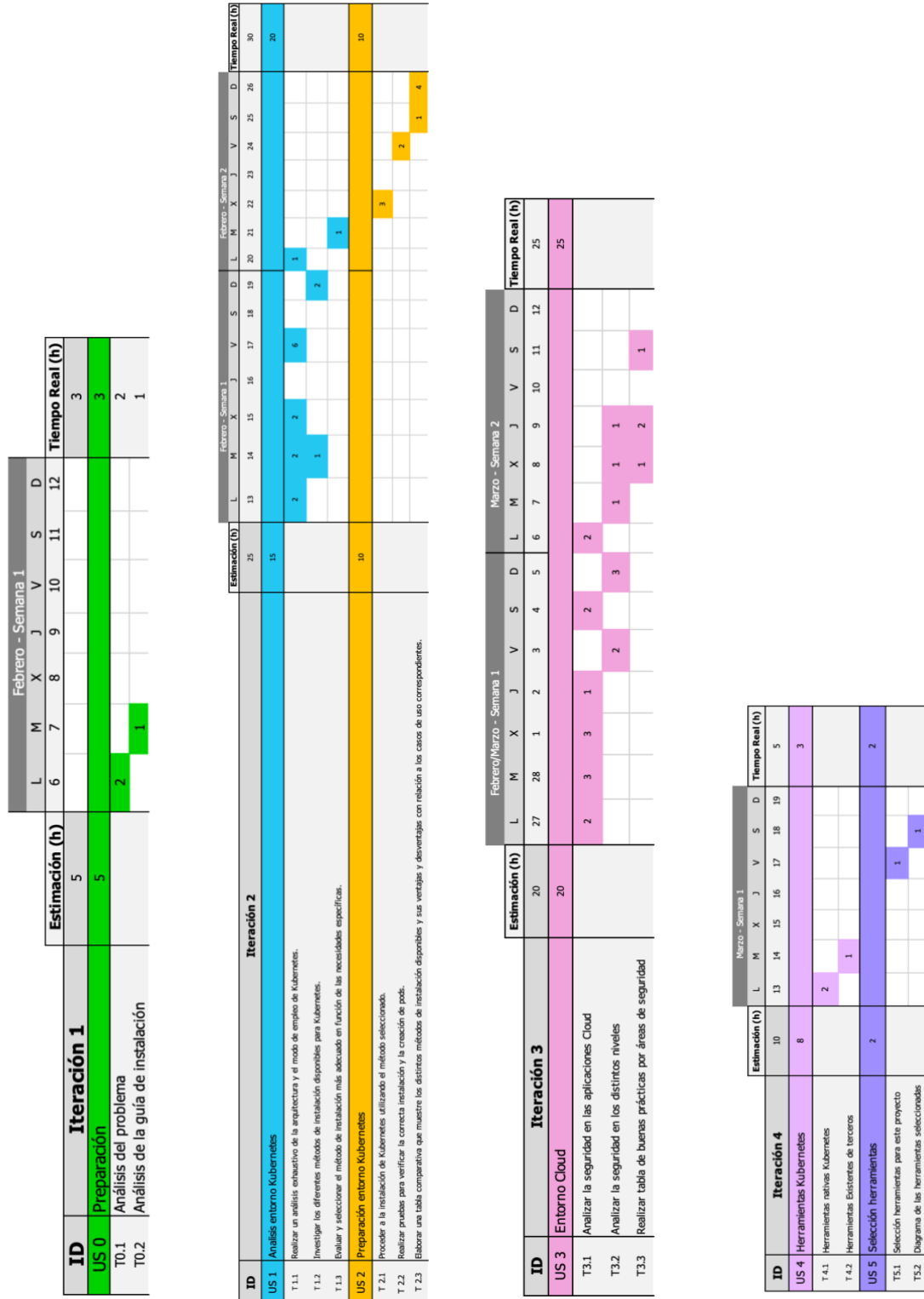
ID: 13	Título: Comparativa herramientas Falco, Sysdig Secure, StackRox	
Descripción: Se analiza y selecciona las características interesantes para comparar las siguientes herramientas: Falco, StackRox y Sysdig Secure. Este proceso incluye la creación de una tabla comparativa.		
Prioridad: Media	Riesgo: Alto	Esfuerzo: 15 h

ID: 14	Título: Guía de instalación	
Descripción: Dada la necesidad de mejorar la eficacia de instalación de las herramientas para el equipo, se redacta una guía que explique la instalación de las herramientas seleccionadas, así como una serie de buenas prácticas a la hora de configurar las herramientas.		
Prioridad: Media	Riesgo: Alto	Esfuerzo: 15 h

ID: 15	Título: Preparación Demostración	
Descripción: Es necesario demostrar cómo proteger un entorno de Kubernetes con algunas de las herramientas mencionadas. Este proceso incluye la selección de las herramientas y la preparación de un guion.		
Prioridad: Media	Riesgo: Medio	Esfuerzo: 20 h

Anexo 3: Iteraciones

En el siguiente anexo, se presentan de manera detallada las iteraciones mencionadas en la sección de metodología y que fueron parcialmente mostradas en la ilustración 7.



ID	Iteración 5	Estimación (h)	Marzo - Semana 1							Marzo/Abril - Semana 2							Tiempo Real (h)
			L	M	X	J	V	S	D	L	M	X	J	V	S	D	
		30	20	21	22	23	24	25	26	27	28	29	30	31	1	2	33
US 6	Herramientas a nivel de cluster	25															27
T6.1	Investigación de las herramientas		2	3	2	1	2	1									
T6.2	Instalación de las herramientas							1	2	1	1		2				
T6.3	Configurar funcionamiento de las herramientas								1	2		2	2	2			
US 7	Tabla comparativa - nivel clúster	5															6
T7.1	Tablas comparativas por sección de las herramientas								1	1				2	1		

ID	Iteración 6	Estimación (h)	Abril - Semana 1							Abril - Semana 2							Tiempo Real (h)
			L	M	X	J	V	S	D	L	M	X	J	V	S	D	
		20	3	4	5	6	7	8	9	10	11	12	13	14	15	16	25
US 8	Herramientas a nivel de tiempo de ejecución	18															22
T8.1	Investigación de las herramientas		2	3	3												
T8.2	Instalación de las herramientas					2	2	2	1	2							
T8.3	Configurar funcionamiento de las herramientas											2		2	1		
US 9	Tabla comparativa - Tiempo de ejecución	2															3
T9.1	Tabla comparativa sección escaneo de imágenes														2	1	

ID	Iteración 7	Estimación (h)	Abril - Semana 1							Abril - Semana 2							Tiempo Real (h)
			L	M	X	J	V	S	D	L	M	X	J	V	S	D	
		30	17	18	19	20	21	22	23	24	25	26	27	28	29	30	35
US 10	Análisis de la Herramienta Falco	25															27
T10.1	Análisis Falco - funcionamiento herramienta		2	1	2	1	1										
T10.2	Instalar Falco					2	3										
T10.3	Configurar Falco							2	2	2	2	2	2	2			
T10.4	Buenas prácticas Falco													1	2		
US 14	Manual de usuario	5															8
T14.1	Redactar Instalación Falco - Manual de usuario					2	2							1	1		
T14.2	Redactar buenas prácticas Falco - Manual de usuario															2	

ID	Iteración 8	Estimación (h)	Mayo - Semana 1							Mayo - Semana 2							Tiempo Real (h)
			L	M	X	J	V	S	D	L	M	X	J	V	S	D	
		25	1	2	3	4	5	6	7	8	9	10	11	12	13	14	20
US 11	Análisis StackRox	20															15
T11.1	Análisis StackRox - funcionamiento herramienta		1	1	1	2	1	3	1								
T11.2	Buenas prácticas StackRox													2	2	1	
US 14	Manual de usuario	5															5
T14.3	Redactar Buenas prácticas StackRox - Manual Usuario													2	2	1	

ID	Iteración 9	Estimación (h)	Mayo - Semana 1							Tiempo Real (h)
			L	M	X	J	V	S	D	
		25	15	16	17	18	19	20	21	10
US 12	Análisis Sysdig Secure	20								7
T12.1	Análisis Sysdig Secure - funcionamiento herramienta		2	2	1					
T12.2	Buenas prácticas Sysdig Secure					1	1			
US 14	Manual de usuario	5								3
T14.3	Redactar Buenas prácticas Sysdig Secure - Manual Usuario							2	1	

ID	Iteración 10	Estimación (h)	Mayo - Semana 1							Tiempo Real (h)
			L	M	X	J	V	S	D	
		25	22	23	24	25	26	27	28	10
US 13	Comparativa herramientas Falco, Sysdig Secure, StackRox	15								10
T13.1	Analizar y seleccionar características interesantes para comparar las herramientas		2	1	2	2				
T13.2	Crear tabla comparativa de las herramientas						2	1		

ID	Iteración 11	Estimación (h)	Mayo/ Junio - Semana 1							Junio - Semana 2							Tiempo Real (h)
			L	M	X	J	V	S	D	L	M	X	J	V	S	D	
		15	29	30	31	1	2	3	4	5	6	7	8	9	10	11	25
US 14	Manual de usuario	15															25
T14.4	Terminar de redactar					2	3	3	2		1	3	1	3	3	4	

ID	Iteración 12	Estimación (h)	Junio - Semana 1							Junio - Semana 2							Tiempo Real (h)
			L	M	X	J	V	S	D	L	M	X	J	V	S	D	
		20	12	13	14	15	16	17	18	19	20	21	22	23	24	25	29
US 15	Preparación Demostración	20															29
T15.1	Selección herramientas para la demo			1	1												
T15.2	Preparar demo con las herramientas					2	3	4	2	1							
T15.3	Guión para la demo						3	3		2	1	2	3	1			

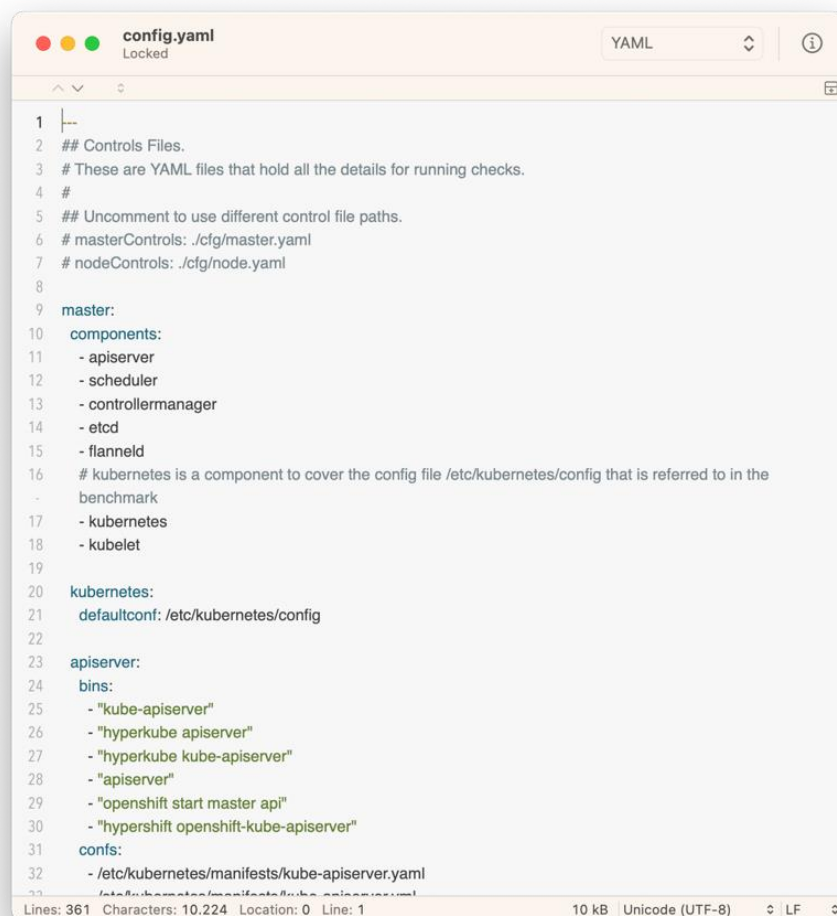
Anexo 4: Código YAML

A continuación, se muestran los ejemplos utilizados a la largo del proyecto:

4.1 A nivel de clúster:

4.1.1 Compliance

- Kube-bench



```
1 |--
2 ## Controls Files.
3 # These are YAML files that hold all the details for running checks.
4 #
5 ## Uncomment to use different control file paths.
6 # masterControls: ./cfg/master.yaml
7 # nodeControls: ./cfg/node.yaml
8
9 master:
10 components:
11 - apiserver
12 - scheduler
13 - controllermanager
14 - etcd
15 - flannel
16 # kubernetes is a component to cover the config file /etc/kubernetes/config that is referred to in the
17 benchmark
18 - kubernetes
19 - kubelet
20 kubernetes:
21 defaultconf: /etc/kubernetes/config
22
23 apiserver:
24 bins:
25 - "kube-apiserver"
26 - "hyperkube apiserver"
27 - "hyperkube kube-apiserver"
28 - "apiserver"
29 - "openshift start master api"
30 - "hypershift openshift-kube-apiserver"
31 confs:
32 - /etc/kubernetes/manifests/kube-apiserver.yaml
33 - /etc/kubernetes/manifests/kube-apiserver.yaml
```

Ilustración 27. *Config.yaml instalación Kube-Bench*

En la ilustración 27, se presenta un extracto del archivo config.yaml utilizado en el proyecto para la creación de los *Pods* necesarios para la instalación de Kube-bench. Para acceder al código completo, se encuentra disponible en mi repositorio de *GitHub*, así como en la documentación oficial de Kube-bench. [27]

4.1.2 Gestión de secretos

La principal función de los ejemplos de esta sección es analizar las herramientas y ver las diferentes opciones que ofrecen para poder así hacer un análisis sencillo y rápido del funcionamiento.

- Sealed Secrets

```

1  apiVersion: bitnami.com/v1alpha1
2  kind: SealedSecret
3  metadata:
4    creationTimestamp: null
5    name: my-secret
6    namespace: default
7  spec:
8    encryptedData:
9      password:
10     AgBFVbJyXqejDQ2og8m76DI+Ysg6E6a6coteRnK+SztlcZEK9IQ76wSOy1nsiXfeONjpcmqwOW7SrJ1OvS/
11     bTzAL1KFxUt+UVkQAwZyX5PM5eriShKy7vn+DC3ZsDPcrHcoAv3v8eW+fyDtrrfqOZH12PVhtPbWD/
12     oopXAluytd8pww67kv5y67RLn2UWUN8ugpBScEVGlg5THSGBILClGw2P1GmRNCTVE/
13     gxdxiuidPWvGixvKHyYsn+AK5R3uKCGctkCy/
14     K5Rwa3cc2v78BxZdleZonVZb52LgQ77DfzUsY9RDsK23C0G57SPW6/RW/
15     DTzpUGvRI03yXjmsBIVWRFqDVFMEhMGATenh9L8NXq6UisLbXLoPi29YZlywy+9r53boUyCvduEyRs87tI3Xlar
16     9qEZr/
17     L4D3ewzb2qF4o8oNW4YWMAXbu9iilQUSUfVrhTg7qgfov7AHkct+uESGgpx891gDC4J8AOJQjqsDQPldCTgZGh
18     OIMQU/GbSBN8M27Xyfos+P/6FWWhAZLLBwVSiepejx1JSffRttu4vc1E5mtTmS15gZsqHp/
19     OIUJvUnYnBQa4oGQdz9Q4zCQicQz6TWxL5+zBuo4TnqocWHBoSGGYa8eBq/
20     tEb7Hf5xy6nkhFHrI7Q4xPK52hfHGZIB9hWU5Mlnvph7XPu5Cjdbiq2KZZ+wXHutiFKFP2R2zFFTs=
21     username: AgClhh9taUHCmajz2qeW7mg888Dg8xeSjqM8wy1FIB7GjMek3xwcnmK7J7VclsA3QXkT9Ql/
22     E1eJOHhqsEeV4WfYYZEw+KvgjikKV/!ss0trp+QytQebHdpeABITxoq4os5xE9e8KPKQ6JS+!/
23     CKMW1oJxe9J3ko6A9aa9hvVM7woG0V11r32PMNyzobeuzrOfnwQ/
24     APjsrIQiia1db95G7QFRiPr15v3sCW4HvnshLdAtU3YdQnu7M2lJ7hjlgWdytuXTeWZakfoANQSwge3ASVUTexBL
25     aqLaJu58APqjFK85bgbFgrJHSuxWqu/yZSeH87PpwwqJ6oV7V3nZYszl0rYsuOCi/
26     vFSii1cb7E3FCmMg7XWlIpCpTOPNX5ozUAKQFEyJ6K2njZw/kr7eSWxbs/HmwnjCXCgcl/
27     UG02Ffpka09cjfR7ZkF6vgEICLpsgi+1tw8VMi8UFDKZKOYyx5PTbj1/
28     nwdz9DN+D2qZOX0TKnC3wTEiTeiciP8Gk+jT0wmrSRWa1vSm9OBFoEMz3RzMxxdtIkb8doSCvoh6uYXB2kbFyr6
29     PZGPiBGMlyRSiWrs23cSvOj1++591d1GQnbeQbdAIW7xflZj4m3+YJyQn8Zd6w3r4352T/yUhdh4jYXAxSJ/
30     MVj0uTrOUdiLBIFbpv8zyijbulRDOVJLSWFnJ7hmb17cut1gXfpbEEkhH/33Y4=
31
32  template:
33    metadata:
34      creationTimestamp: null
35      name: my-secret
36      namespace: default
37
38

```

Ilustración 28. Sealed-secret.yaml file

A la hora de ejecutar el comando de creación de secreto se genera un archivo como la ilustración 28 donde aparece la contraseña cifrada. [28]

4.1.3 Gestión de políticas

- Kyverno
 - Tipo de política: *Generate*

```

1  apiVersion: kyverno.io/v1
2  kind: ClusterPolicy
3  metadata:
4    name: add-networkpolicy
5  annotations:
6    policies.kyverno.io/title: Add Network Policy
7    policies.kyverno.io/category: Multi-Tenancy, EKS Best Practices
8    policies.kyverno.io/subject: NetworkPolicy
9    policies.kyverno.io/minversion: 1.6.0
10   policies.kyverno.io/description: >-
11   By default, Kubernetes allows communications across all Pods within a cluster.
12   The NetworkPolicy resource and a CNI plug-in that supports NetworkPolicy must be used to restrict
13   communications. A default NetworkPolicy should be configured for each Namespace to
14   default deny all ingress and egress traffic to the Pods in the Namespace. Application
15   teams can then configure additional NetworkPolicy resources to allow desired traffic
16   to application Pods from select sources. This policy will create a new NetworkPolicy resource
17   named 'default-deny' which will deny all traffic anytime a new Namespace is created.
18  spec:
19    rules:
20      - name: default-deny
21        match:
22          any:
23            - resources:
24              kinds:
25                - Namespace
26        generate:
27          apiVersion: networking.k8s.io/v1
28          kind: NetworkPolicy
29          name: default-deny
30          namespace: "{{(request.object.metadata.name)}}"
31          synchronize: true
32          data:
33            spec:
34              # select all pods in the namespace
35              podSelector: {}
36              # deny all traffic

```

Ilustración 29. Política de tipo "Generate": Añadir políticas de red

En la ilustración 29, se establece una política que generará un recurso de NetworkPolicy denominado *default-deny*, el cual tendrá la función de rechazar todo el tráfico en cada ocasión en que se cree un nuevo Namespace. El código completo se encuentra en el repositorio de GitHub o en la documentación oficial [47].



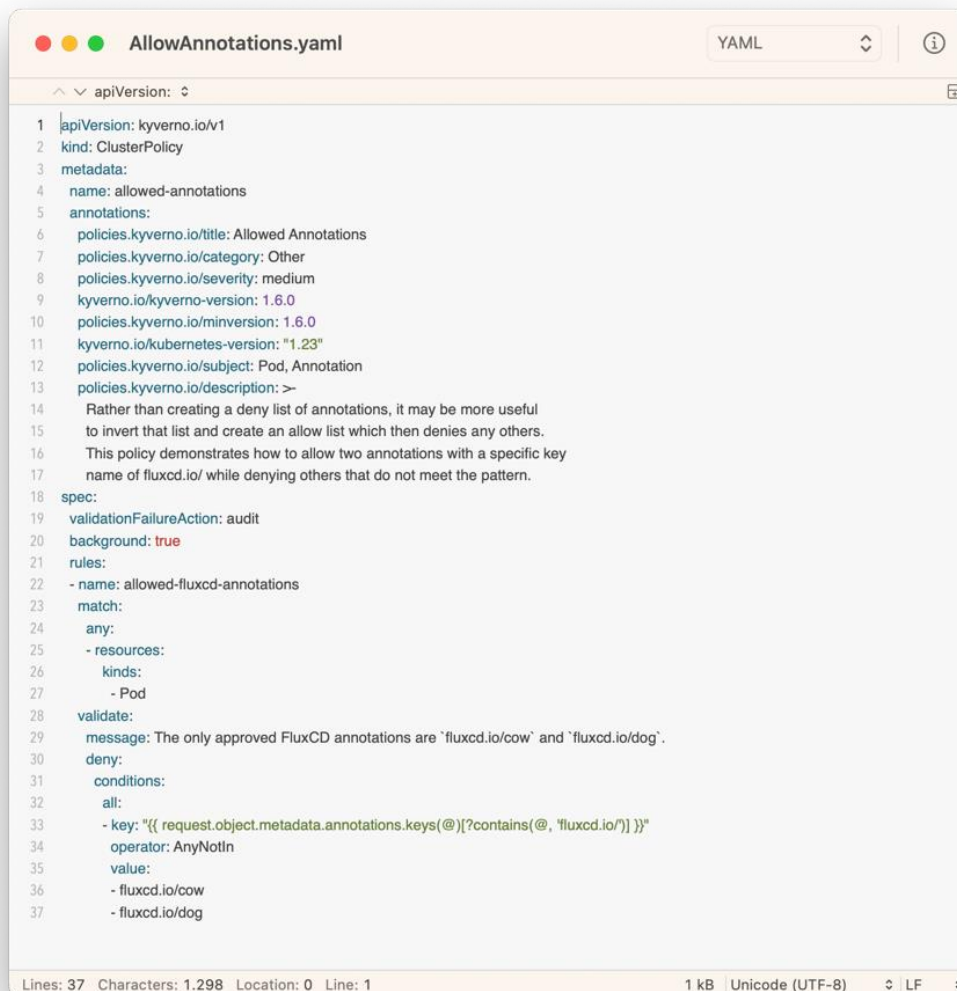
- Tipo de política: *Mutate*

```
1  apiVersion: kyverno.io/v1
2  kind: ClusterPolicy
3  metadata:
4    name: add-labels
5  annotations:
6    policies.kyverno.io/title: Add Labels
7    policies.kyverno.io/category: Sample
8    policies.kyverno.io/minversion: 1.6.0
9    policies.kyverno.io/severity: medium
10   policies.kyverno.io/subject: Label
11   policies.kyverno.io/description: >-
12     Labels are used as an important source of metadata describing objects in various ways
13     or triggering other functionality. Labels are also a very basic concept and should be
14     used throughout Kubernetes. This policy performs a simple mutation which adds a label
15     `foo=bar` to Pods, Services, ConfigMaps, and Secrets.
16  spec:
17    rules:
18    - name: add-labels
19      match:
20        any:
21        - resources:
22            kinds:
23              - Pod
24              - Service
25              - ConfigMap
26              - Secret
27        mutate:
28          patchStrategicMerge:
29            metadata:
30              labels:
31                foo: bar
```

Ilustración 30. Política de tipo "Mutate": Añadir etiquetas

En la ilustración 30, se muestra una política que hace referencia a las etiquetas. Se ejecuta una modificación sencilla que añade una etiqueta foo=bar a los pods, services, configMaps y secrets. El código se encuentra en el repositorio de GitHub o en la documentación oficial. [48]

- Tipo de política: *Validate*

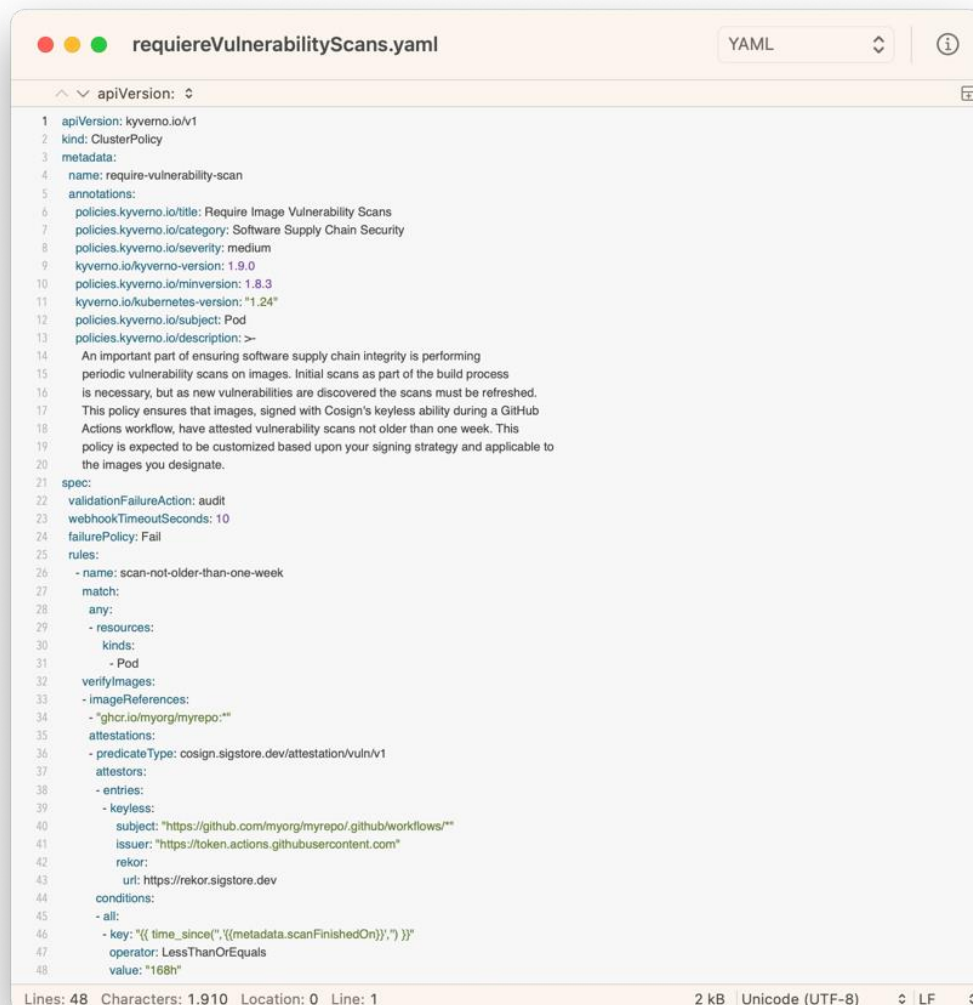


```
1 apiVersion: kyverno.io/v1
2 kind: ClusterPolicy
3 metadata:
4   name: allowed-annotations
5 annotations:
6   policies.kyverno.io/title: Allowed Annotations
7   policies.kyverno.io/category: Other
8   policies.kyverno.io/severity: medium
9   kyverno.io/kyverno-version: 1.6.0
10  policies.kyverno.io/minversion: 1.6.0
11  kyverno.io/kubernetes-version: "1.23"
12  policies.kyverno.io/subject: Pod, Annotation
13  policies.kyverno.io/description: >
14    Rather than creating a deny list of annotations, it may be more useful
15    to invert that list and create an allow list which then denies any others.
16    This policy demonstrates how to allow two annotations with a specific key
17    name of fluxcd.io/ while denying others that do not meet the pattern.
18 spec:
19   validationFailureAction: audit
20   background: true
21   rules:
22   - name: allowed-fluxcd-annotations
23     match:
24       any:
25       - resources:
26         kinds:
27         - Pod
28     validate:
29       message: The only approved FluxCD annotations are `fluxcd.io/cow` and `fluxcd.io/dog`.
30       deny:
31         conditions:
32         all:
33         - key: "{{ request.object.metadata.annotations.keys(@)?contains(@, 'fluxcd.io/')}}"
34           operator: AnyNotIn
35           value:
36           - fluxcd.io/cow
37           - fluxcd.io/dog
```

Ilustración 31. Política de tipo "Validate": Permite anotaciones

En la ilustración 31, se muestra una política la cual permite dos anotaciones con un nombre clave específico de fluxcd.io/, al tiempo que se deniegan aquellas que no cumplen con dicho patrón. El código se encuentra en el repositorio de GitHub o en la documentación oficial [49].

- Tipo de política: *Verify Images*

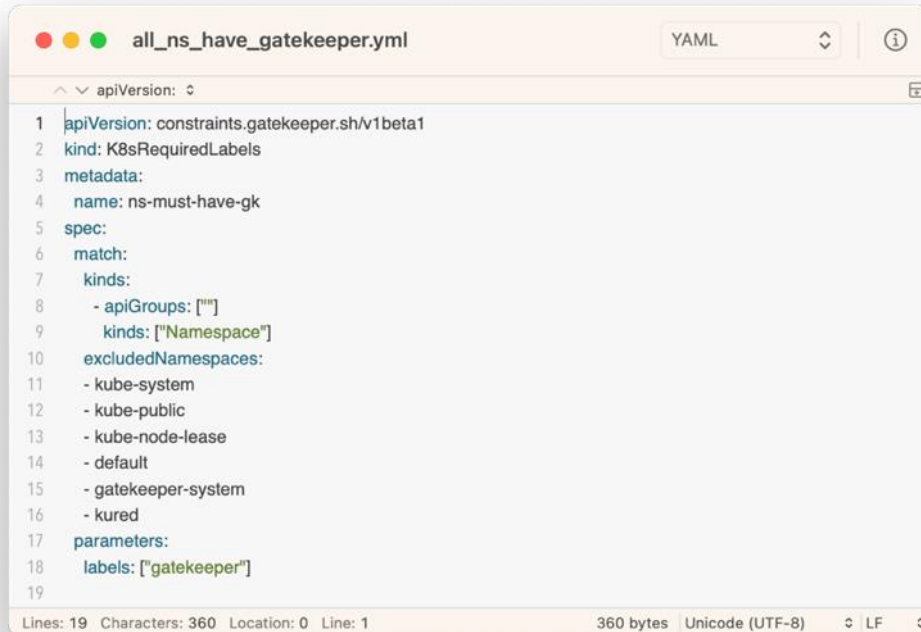


```
1 apiVersion: kyverno.io/v1
2 kind: ClusterPolicy
3 metadata:
4   name: require-vulnerability-scan
5 annotations:
6   policies.kyverno.io/title: Require Image Vulnerability Scans
7   policies.kyverno.io/category: Software Supply Chain Security
8   policies.kyverno.io/severity: medium
9   kyverno.io/kyverno-version: 1.9.0
10  policies.kyverno.io/minversion: 1.8.3
11  kyverno.io/kubernetes-version: "1.24"
12  policies.kyverno.io/subject: Pod
13  policies.kyverno.io/description: >-
14  An important part of ensuring software supply chain integrity is performing
15  periodic vulnerability scans on images. Initial scans as part of the build process
16  is necessary, but as new vulnerabilities are discovered the scans must be refreshed.
17  This policy ensures that images, signed with Cosign's keyless ability during a GitHub
18  Actions workflow, have attested vulnerability scans not older than one week. This
19  policy is expected to be customized based upon your signing strategy and applicable to
20  the images you designate.
21 spec:
22   validationFailureAction: audit
23   webhookTimeoutSeconds: 10
24   failurePolicy: Fail
25   rules:
26   - name: scan-not-older-than-one-week
27     match:
28       any:
29       - resources:
30         kinds:
31         - Pod
32       verifyImages:
33       - imageReferences:
34         - "ghcr.io/myorg/myrepo:*"
35         attestations:
36         - predicateType: cosign.sigstore.dev/attestation/vuln/v1
37         attestors:
38         - entries:
39           - keyless:
40             subject: "https://github.com/myorg/myrepo/github/workflows/"
41             issuer: "https://token.actions.githubusercontent.com"
42             rekor:
43               uri: https://rekor.sigstore.dev
44         conditions:
45         - all:
46         - key: "({{ time_since('', {{metadata.scanFinishedOn}}, '') }}"
47           operator: LessThanOrEquals
48           value: *168h
```

Ilustración 32. Política de tipo "Verify Images": Requiere de escaneo de vulnerabilidades

En esta ilustración 32, se muestra una política que tiene como objetivo asegurar que las imágenes, firmadas con la capacidad sin clave de Cosign durante un flujo de trabajo de GitHub Actions, cuenten con escaneos de vulnerabilidad atestados que no superen una antigüedad de una semana. Esta se puede adaptar según su estrategia de firma y que sea aplicable a las imágenes que sea se designe. El código se encuentra en mi repositorio de GitHub o en la documentación oficial [50].

- Open Policy Agent – GateKeeper



```
1 apiVersion: constraints.gatekeeper.sh/v1beta1
2 kind: K8sRequiredLabels
3 metadata:
4   name: ns-must-have-gk
5 spec:
6   match:
7     kinds:
8     - apiGroups: [""]
9       kinds: ["Namespace"]
10  excludedNamespaces:
11    - kube-system
12    - kube-public
13    - kube-node-lease
14    - default
15    - gatekeeper-system
16    - kured
17  parameters:
18    labels: ["gatekeeper"]
19
```

Ilustración 33. Ejemplo de política en Gatekeeper

En la ilustración 33, se presenta un ejemplo de política creada en OPA. Se observa la necesidad de crear un archivo YAML para crear. En este ejemplo se crea una política en la que todos los pods necesitan tener Gatekeeper instalado. [51]



Anexo 5: Guía de instalación

La guía de instalación se adjunta con las memorias en formato PDF.



Ilustración 34. *Guía de instalación para macOS*