

**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

**Zaragoza, 25 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 **7 julio 2023**



---

## Dedicatoria y Agradecimientos

## Tabla de contenido

<b>Tabla de ilustraciones .....</b>	<b>8</b>
<b>Índice de tablas .....</b>	<b>9</b>
<b>Resumen .....</b>	<b>10</b>
<b>Abstract.....</b>	<b>10</b>
<b>1.     Introducción .....</b>	<b>11</b>
<b>PArox. 3 paginas FALTA .....</b>	<b>11</b>
<b>2.     Antecedentes / Estado del Arte.....</b>	<b>12</b>
<b>1.1.   De sistemas monolíticos a microservicios .....</b>	<b>12</b>
<b>1.2.   Contenedores y Orquestación de contenedores.....</b>	<b>14</b>
1.2.1.   Herramientas existentes .....	15
Docker [6].....	15
Kubernetes [7].....	16
1.2.2.   DevOps y Soluciones de seguridad en el mercado actual.....	16
<b>3.     Objetivos .....</b>	<b>19</b>
<b>4.     Metodología .....</b>	<b>20</b>
<b>4.1   Extreme Programming (XP) .....</b>	<b>20</b>
<b>4.2   Aplicación de la tecnología .....</b>	<b>21</b>
<b>4.3   Seguimiento del desarrollo.....</b>	<b>21</b>
4.3.1   Trello [16] .....	21
4.3.2   Diagrama de trabajo.....	22
<b>5.     Análisis &amp; Implementación.....</b>	<b>24</b>
<b>5.1   Iteración 1: Análisis.....</b>	<b>24</b>
5.1.2   Análisis del problema.....	24
5.1.3   Análisis de la guía de instalación.....	25
<b>5.2   Iteración 2 – Investigación &amp; Análisis de Kubernetes .....</b>	<b>25</b>
<b>5.3   Iteración 3 – Análisis de las características de seguridad en Kubernetes .....</b>	<b>29</b>
5.3.1   Tareas US 3.....	29
5.3.2   Desarrollo.....	29
5.3.3   Resultados.....	35
<b>5.4   Iteración 4 – Investigación de herramientas para proteger un entorno de Kubernetes .....</b>	<b>38</b>
5.4.1   Tareas US 4.....	38
5.4.2   Desarrollo.....	38
5.4.3   Resultados.....	38
<b>5.5   Iteración 5 – Análisis de las herramientas seleccionadas – Cluster layer .....</b>	<b>40</b>
5.5.1   Tareas US 5.....	40
5.5.2   Desarrollo.....	40
5.5.3   Resultados.....	46
<b>5.6   Iteración 6 – Análisis de las herramientas seleccionadas - Container Run time layer</b>	<b>51</b>
5.6.1   Tareas US 6.....	51
5.6.2   Desarrollo.....	51
5.6.3   Resultados.....	51
<b>5.7   Iteración 7 – Análisis Falco .....</b>	<b>52</b>

---

5.7.1	<i>Tareas US 7</i> .....	52
5.7.2	<i>Desarrollo</i> .....	52
5.7.3	<i>Resultados</i> .....	52
<b>5.8</b>	<b>Iteración 8 – Análisis StackRox</b> .....	<b>52</b>
5.8.1	<i>Tareas US 8</i> .....	52
5.8.2	<i>Desarrollo</i> .....	53
5.8.3	<i>Resultados</i> .....	53
<b>5.9</b>	<b>Iteración 9 – Análisis Sysdig Secure</b> .....	<b>53</b>
5.9.1	<i>Tareas US 9</i> .....	53
5.9.2	<i>Desarrollo</i> .....	54
5.9.3	<i>Resultados</i> .....	54
<b>5.10</b>	<b>Iteración 10 – Análisis de las principales herramientas Falco, StackRox, Sysdig</b> <b>54</b>	
5.10.1	<i>Tareas US 10</i> .....	54
5.10.2	<i>Desarrollo</i> .....	54
5.10.3	<i>Resultados</i> .....	54
<b>A.</b>	<b>Estudio económico</b> .....	<b>55</b>
<b>8.1</b>	<b>Desglose de costes</b> .....	<b>55</b>
a.	<i>Costes materiales</i> .....	55
b.	<i>Costes humanos</i> .....	56
c.	<i>Costes de infraestructura</i> .....	56
d.	<i>Costes totales</i> .....	56
<b>B.</b>	<b>Resultados</b> .....	<b>57</b>
<b>C.</b>	<b>Conclusiones</b> .....	<b>58</b>
a.	<b>Propuesta de mejora</b> .....	58
<b>D.</b>	<b>Bibliografía</b> .....	<b>59</b>
<b>E.</b>	<b>Anexos</b> .....	<b>61</b>
a.	<b>Anexo 1: Propuesta</b> .....	61
b.	<b>Anexo 2: Reuniones</b> .....	62
c.	<b>Anexo 3: Historias de Usuario</b> .....	63
d.	<b>Anexo 4: Horas trabajadas</b> .....	64

## Tabla de ilustraciones

Ilustración 1 - <i>Comparativa de Sistema Monolíticos y Microsistemas</i> .....	13
Ilustración 2 - <i>Comparativa de Máquinas virtuales y Contenedores</i> .....	6
Ilustración 3 - <i>Ciclo de desarrollo del Software en DevOps14</i> .....	6
Ilustración 4 - <i>Clasificación de herramientas de seguridad existentes</i> .....	6
Ilustración 5 - <i>Trello´s Dashboard del proyecto</i> .....	
Ilustración 6. Diagrama de Gantt del proyecto .....	
Ilustración 7. Kubernetes dashboard.....	
Ilustración 8. Arquitectura Kubernetes .....	
Ilustración 10. Selección de herramientas a analizar	

## Índice de tablas

Tabla 1 - Comparativa de métodos de instalación Kubernetes.....9

Tabla 2. Características de seguridad en Kubernetes

## 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 dos herramientas de seguridad para K8s: Falco y StackRox. Falco es una herramienta de prevención de intrusiones que se integra directamente con Kubernetes y se enfoca en detectar comportamientos inusuales en tiempo real. Por otro lado, StackRox es una plataforma de seguridad para Kubernetes que se enfoca en la administración de políticas de seguridad y cumplimiento, detección de amenazas y respuesta automatizada.

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 Falco y StackRox 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, StackRox, Falco, Seguridad, microservicios*

## Abstract

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

Security at Kubernetes is crucial as a security breach can have serious consequences such as data loss or service interruption, which would damage the organization's reputation. Kubernetes environments are highly dynamic and scalable, making them vulnerable to security threats. Updated and effective tools are needed to detect and mitigate security risks in the cloud in a timely manner. Therefore, this project aims to analyze two security tools for K8s: Falco and StackRox. Falco is an intrusion prevention tool that integrates directly with Kubernetes and focuses on detecting unusual behaviors in real-time. On the other hand, StackRox is a security platform for Kubernetes that focuses on security policy management and compliance, threat detection, and automated response.

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

**Keywords:** *Kubernetes, StackRox, Falco, Security, microservices*

## 1. Introducción

PArox. 3 paginas FALTA

"¿Cuál es el problema?"  
*Debe centrar al lector en la problemática del Proyecto desarrollado. Debería empezar contextualizando la importancia del proyecto dentro de una problemática general que se describa brevemente. La introducción debería terminar indicando la organización del proyecto para acometer el proyecto a desarrollar.*

En la actualidad, la tecnología de contenedores se ha convertido en una herramienta fundamental para el despliegue de aplicaciones y servicios en la nube. En este contexto, Kubernetes se ha consolidado como la plataforma de orquestación de contenedores más popular y ampliamente utilizada. Sin embargo, la seguridad en Kubernetes es un tema crítico que ha recibido mucha atención en los últimos años.

El problema radica en que Kubernetes, al igual que cualquier otra tecnología, tiene vulnerabilidades y debilidades que pueden ser explotadas por atacantes malintencionados. La naturaleza distribuida y escalable de Kubernetes también lo hace más complejo de proteger y monitorear adecuadamente. Además, dado que Kubernetes es utilizado en entornos empresariales críticos, cualquier violación de la seguridad puede tener consecuencias graves.

Es por ello que resulta crucial desarrollar estrategias y herramientas de seguridad para proteger adecuadamente las implementaciones de Kubernetes. De esta manera, se pueden minimizar los riesgos de ataques, fugas de información o interrupciones de servicio, asegurando así la integridad y disponibilidad de los sistemas.

Para abordar esta problemática, el proyecto que se desarrollará tiene como objetivo investigar y comprar de forma exhaustiva las herramientas existentes desde el nivel de clúster hasta el nivel de aplicación, tanto en términos de plataforma como de código.

El proyecto se lleva a cabo en colaboración con la empresa NTTDATA para mejorar la seguridad de los datos en la nube en los entornos de Kubernetes.

CAPTURA ATQAUES – Kaspersky

---

## 2. Antecedentes / 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.

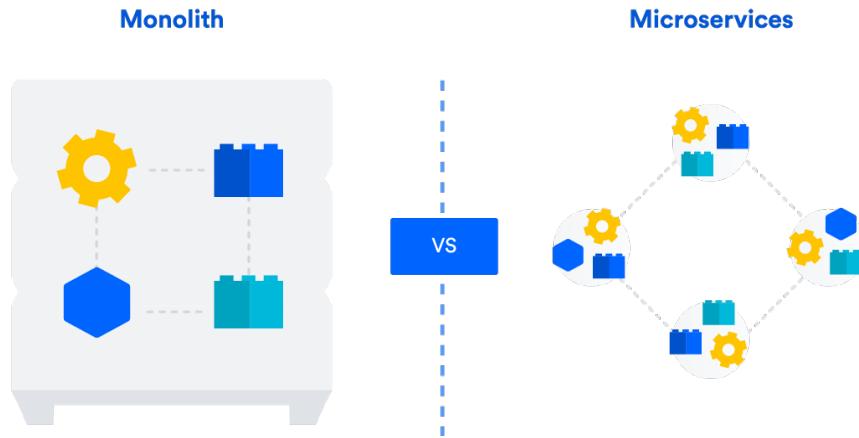
### 1.1. De sistemas monolíticos a microservicios

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 [1] 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. [1]



**Ilustración 1.** Comparativa de Sistema Monolíticos y Microsistemas [1]

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 [1], Spotify [2], Adidas [3] o Amazon [4] 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.

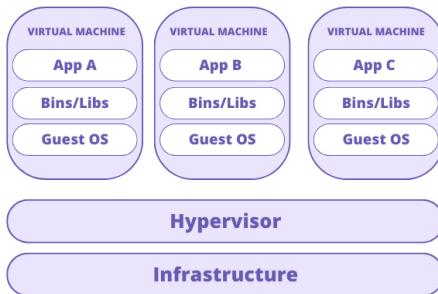
## **1.2. Contenedores y Orquestación de contenedores**

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.

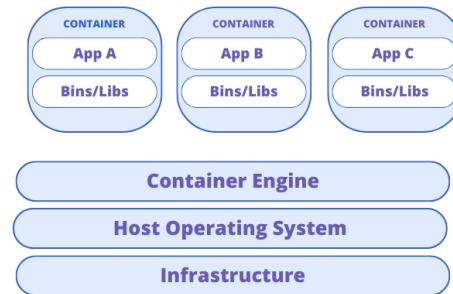
Los contenedores son una forma de virtualización del sistema operativo que permite utilizar un único contenedor para ejecutar todo, desde microservicios o procesos de software hasta grandes aplicaciones. Contienen todos los archivos ejecutables, códigos binarios, bibliotecas y archivos de configuración 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. [5]

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.



Virtual Machines



Containers

### **Ilustración 2. Comparativa de Máquinas virtuales y Contenedores [5]**

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, 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. [5]

#### *1.2.1. Herramientas existentes*

##### *Docker [6]*

Docker 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, podemos implementar y escalar rápidamente sus aplicaciones a cualquier entorno con la confianza de que su código se ejecutará en cualquier lugar. Docker es una plataforma de software que permite la creación de contenedores para aplicaciones y servicios, lo que facilita el desarrollo, la implementación y la gestión de aplicaciones en cualquier entorno. 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 [7]*

Kubernetes 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 Fundación. 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 [8], 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 [9] 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 [10], 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 [11], 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.

#### *1.2.2. DevOps y Soluciones de seguridad en el mercado actual*

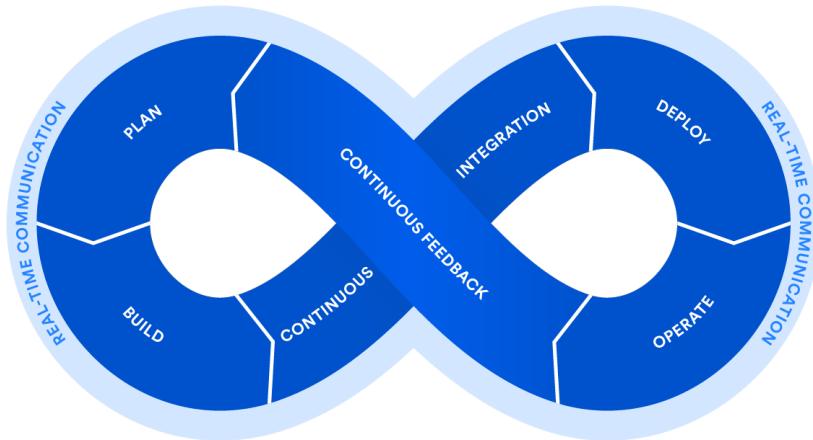
Las empresas tienen la responsabilidad de mantener el sistema operativo actualizado e instalar los parches de seguridad que sean necesarios en todo momento. Al igual que ocurre en los servidores que son propiedad de una empresa, es necesario mantener políticas de seguridad tradicionales como el control 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 nos han acompañado 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 de la ciberseguridad.

En un entorno DevOps [12] 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.

Para este cometido han surgido los perfiles DevSecOps [13], que se centran en añadir seguridad al ciclo de vida de software tanto en el despliegue como monitorización de los sistemas.

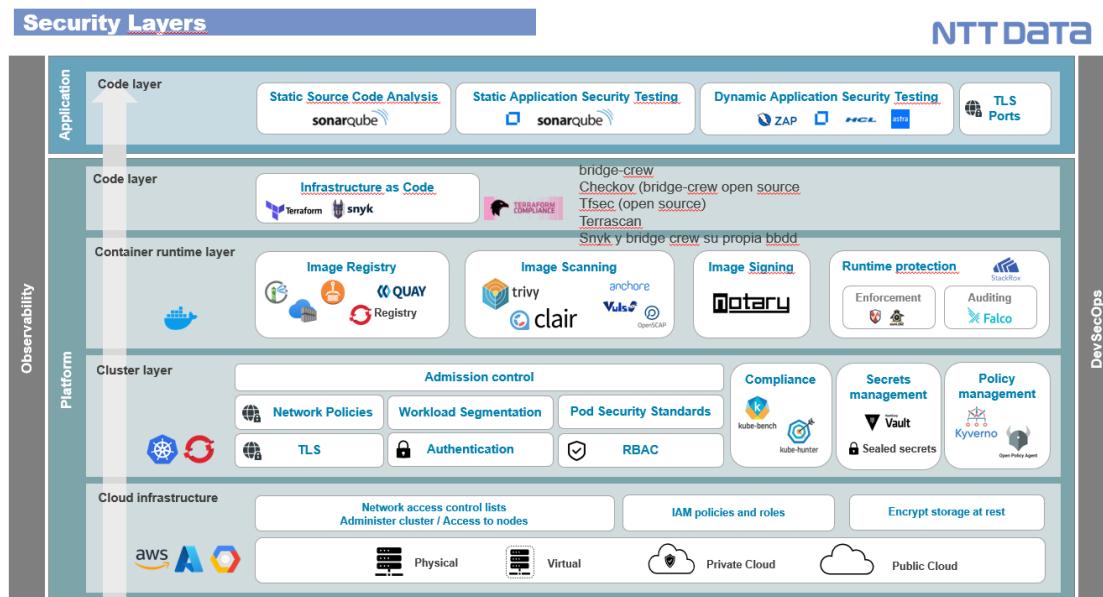


**Ilustración 3.** Ciclo de desarrollo del Software en DevOps [14]

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.

Esta es una clasificación de herramientas de seguridad existentes por categorías.



**Ilustración 4.** Clasificación de herramientas de seguridad existentes

En este proyecto mencionaremos la importancia de los niveles de seguridad y cuáles son las más empleadas además de una breve comparativa. Y nos focalizaremos en el apartado de *Runtime protection* analizando en profundidad las herramientas como Falco y StackRox.

### **3. Objetivos**

Los objetivos principales que se definieron para este proyecto, 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: Falco y StackRox
- 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, **XXX horas** de trabajo a desarrollar por, en este caso, una sola persona. La fecha de entrega de este proyecto es el **25 de junio de 2023**, por lo que la flexibilidad para la iteración es limitada.

El proyecto se combina con otras asignaturas y trabajos, 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, estimo 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 [15] (XP) como metodología para este proyecto.

##### 4.1 Extreme Programming (XP)

Extreme Programming es una metodología de desarrollo de software ágil [15] 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 refactorización del código y la retroalimentación [15], 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 el Objetivo 2 del proyecto, que implica un estudio comparativo de dos herramientas, ya que hasta que no se hayan analizado 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 del equipo 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 [15] y altamente cambiantes, así como para aquellos con un alto riesgo técnico, como puede ser el caso de la falta de experiencia en el desarrollo de software para este proyecto.

#### **4.2 Aplicación de la tecnología**

XP, al igual que la mayoría de las técnicas ágiles [15], 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. 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 de 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 [16]**

Para realizar el seguimiento se ha empleado la herramienta Trello. He 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 puedes crear, mover eliminar tarjetas, de esta forma conseguimos de forma visual conocer el estado actual de nuestro proyecto además de conseguir mayor eficacia en cuanto a tiempo empleado en la herramienta.

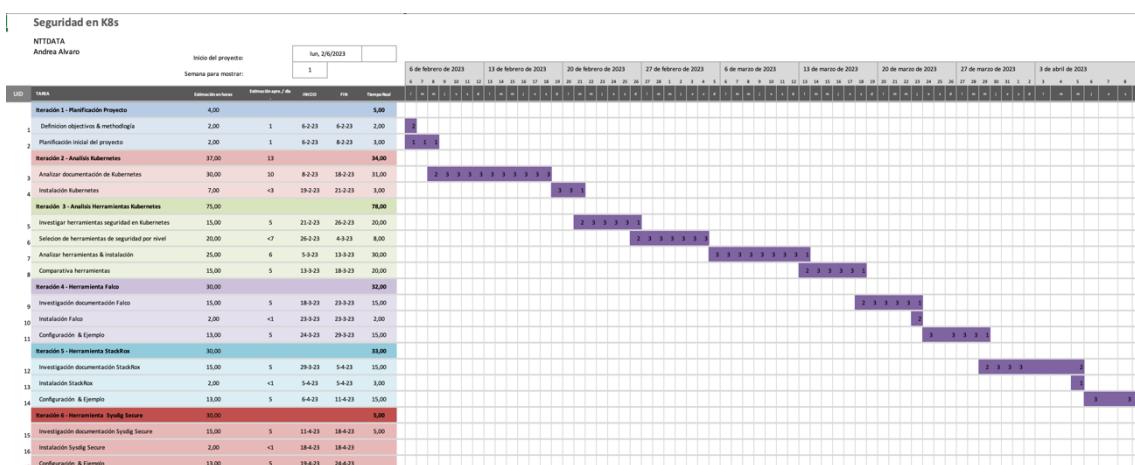
En nuestro caso, al tratarse de un proyecto grande y al trabajar por iteraciones, trabajaremos de la siguiente forma. Al inicio de cada iteración las historias del usuario se dividirán en tareas más pequeñas, cada tarea será una tarjeta y se añadirá una estimación y una prioridad. A medida que

se vayan completando las tareas y desarrollando la iteración las tarjetas se irán cambiando de columnas o estado hasta ser completadas.

**Ilustración 5.** 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<sup>17</sup>. 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. **SUPRIMIR ITERACION 1**



**Ilustración 6.** Diagrama de Gantt del proyecto

La Ilustración 6 muestra una estimación del tiempo por iteración durante el periodo disponible para el proyecto. Con el fin de organizar y gestionar adecuadamente el trabajo, se ha optado por dividir el proyecto en varias iteraciones, cada una representada por un color diferente en la Ilustración 6. Cada iteración se divide en diferentes actividades que se llevarán a cabo.

Dado que el enfoque principal del proyecto es el análisis de herramientas, existe la posibilidad de realizar las iteraciones de forma simultánea o en secuencia. Sin embargo, se ha decidido llevar a cabo una iteración tras otra, siguiendo un orden establecido. De esta manera, se asegura un flujo lógico y organizado en el desarrollo del proyecto.

**EXPLICAR ITERACIONES**

## 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 exhaustivo 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 1

- 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:

- Aplicación
  - Seguridad de código:
- Plataforma:
  - Seguridad de código:
    - Seguridad del código como Infraestructura
  - 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 empresariales. 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 que, a través del análisis de herramientas, se busca identificar las mejores soluciones disponibles en el mercado para fortalecer la seguridad de Kubernetes.

### **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 2*

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 y 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 que 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 es una plataforma de orquestación de contenedores de código abierto utilizada para automatizar, escalar y administrar aplicaciones en contenedores. Fue desarrollado por Google y luego donado a la Cloud Native Computing Foundation (CNCF). Kubernetes proporciona un entorno para coordinar y administrar la ejecución de contenedores en clústeres de servidores.[21]

---

La principal función de Kubernetes 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.[21]

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 apéndice adjunto, se proporciona una guía detallada de instalación que explica paso a paso cómo instalar Docker en el entorno correspondiente. 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.

Como podemos observar Minikube dispone de Dashboard una forma muy cómoda de tener una amplia visión del estado del trabajo en Kubernetes. Dashboard es una interfaz de usuario basada en web de propósito general para clústeres de Kubernetes. Permite a los usuarios gestionar y solucionar problemas de aplicaciones que se ejecutan en el clúster, así como del propio clúster.

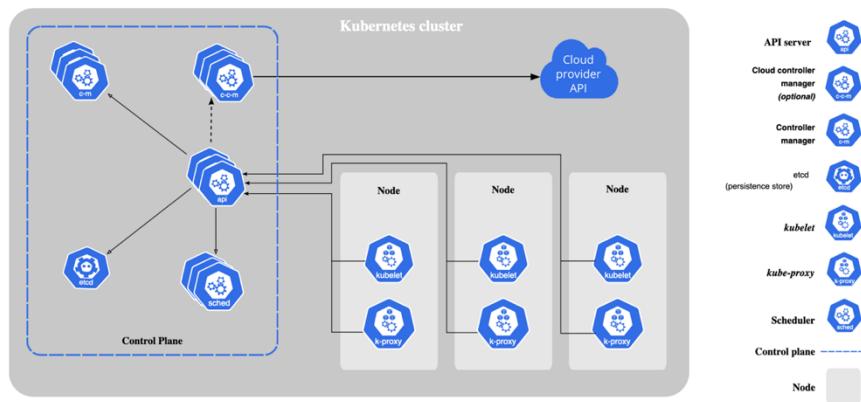


**Ilustración 7.** Kubernetes dashboard

La guía de instalación se encuentra en el *Anexo* 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 suele ejecutarse en múltiples equipos, y un clúster puede contar con varios nodos, brindando así tolerancia a fallos y alta disponibilidad. En la ilustración 9, da una imagen general de los componentes en Kubernetes.



**Ilustración 8.** Arquitectura Kubernetes [22]

### 5.2.3 Resultado

Existen diferentes métodos de instalación aquí se recoge una comparativa de los métodos.

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 mantenable 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 Kubernetes

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

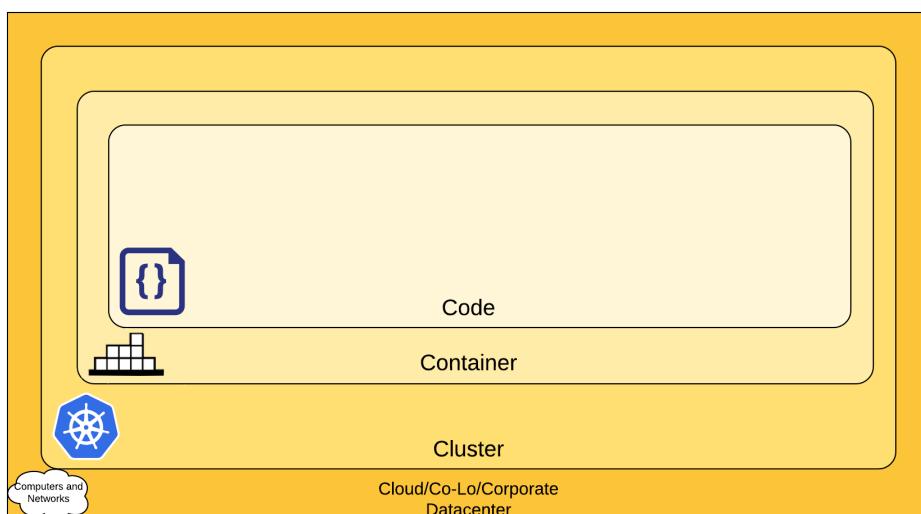
#### 5.3.1 Tareas US 3

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

#### 5.3.2 Desarrollo

##### FALTAN REFERENCIAS

Con el auge del uso de Cloud y los nuevos perfiles surgidos de DevSecOps la seguridad en el Cloud se ha convertido en un desafío. La seguridad en la nube se divide en 4 categorías que observamos en la Ilustración 9.



**Ilustración 9.** Las 4C de la seguridad nativa en la nube [23]

##### a. 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 [24] específicas para asegurar la ejecución segura de cargas de trabajo en su entorno.

Durante mi experiencia en la empresa, he observado 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.

*b. 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 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.

*c. Análisis en seguridad en los clústeres*

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

- **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.

*d. Análisis seguridad en los contenedores*

La seguridad en los contenedores es un tema crítico, y requiere de una atención rigurosa y exhaustiva. Los contenedores, como Docker, han ganado popularidad en el despliegue de aplicaciones debido a su capacidad de encapsular y distribuir aplicaciones de manera eficiente. Sin embargo, su adopción también ha planteado nuevos desafíos en términos de seguridad. Es importante tener en cuenta diversas medidas para garantizar la seguridad de los contenedores. En primer lugar, se debe considerar la selección de imágenes base confiables y actualizadas. Las imágenes utilizadas como base para los contenedores deben provenir de fuentes confiables y ser revisadas en busca de posibles vulnerabilidades conocidas. Además, se debe mantener un proceso de actualización regular para asegurar que los contenedores estén protegidos contra las últimas amenazas de seguridad.

Otro aspecto fundamental es la configuración segura de los contenedores. Esto implica aplicar los principios de mínimo privilegio, lo cual significa limitar los permisos y privilegios del contenedor solo a lo estrictamente necesario para su funcionamiento. Se deben deshabilitar servicios y características innecesarias, y se deben aplicar políticas de acceso y autenticación adecuadas para prevenir accesos no autorizados.

Además, es crucial asegurar la comunicación entre los contenedores y con otros sistemas. Se deben utilizar protocolos de comunicación seguros, como HTTPS, y se debe aplicar cifrado para proteger los datos en tránsito. Asimismo, 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 también es un aspecto 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. Además, se debe mantener un proceso de parcheo y actualización constante para corregir las vulnerabilidades conocidas.

Por último, la educación y concienciación de los desarrolladores y operadores de los contenedores es esencial. Se deben proporcionar pautas de seguridad y mejores prácticas, y se debe fomentar una cultura de seguridad en la que se valoren y se tomen en serio los aspectos relacionados con la seguridad de los contenedores.

#### e. Seguridad en las imágenes Docker

Para este proyecto se ha decidido utilizar Docker por tanto revisamos las características a proteger en las imágenes de Docker. El análisis de seguridad en imágenes Docker desempeña un papel fundamental, al asegurar la integridad y salvaguardar los sistemas en uso. Las imágenes Docker, que facilitan la encapsulación de aplicaciones y servicios en contenedores, pueden presentar vulnerabilidades que los actores maliciosos podrían explotar, lo que comprometería la seguridad de los entornos de ejecución.

Por ello es importante realizar un análisis exhaustivo de las imágenes Docker antes de su implementación. Este proceso comprende una evaluación minuciosa de la imagen en busca de posibles vulnerabilidades conocidas, para lo cual se emplean herramientas y técnicas de análisis automatizado. Dichas herramientas escanean la imagen en procura de vulnerabilidades de software, dependencias obsoletas o configuraciones erróneas.

Adicionalmente, es esencial adherirse a las mejores prácticas a fin de garantizar la integridad de las imágenes Docker. Entre estas prácticas se incluye el uso de imágenes base provenientes de fuentes confiables y actualizadas, así como mantener las imágenes actualizadas con las últimas correcciones de seguridad. Asimismo, es de vital importancia restringir el acceso a las imágenes Docker y aplicar políticas de control de acceso que impidan modificaciones no autorizadas.

La gestión adecuada de las dependencias constituye otro aspecto crítico en las imágenes Docker. Las dependencias utilizadas en la construcción de la imagen deben someterse a una evaluación de seguridad, y se deben tomar medidas para mantenerlas actualizadas con las versiones más seguras disponibles. Esto contribuye a mitigar los riesgos asociados con vulnerabilidades conocidas en las dependencias utilizadas.

---

Por otro lado, como mencionamos anteriormente se deben implementar mecanismos de monitoreo y registro que permitan detectar y responder de manera oportuna a cualquier actividad o comportamiento sospechoso relacionado con las imágenes Docker. Esto puede incluir la configuración de alertas y registros de auditoría para detectar intrusiones o intentos de explotación.

*f. Análisis Seguridad en el código*

Por último, cabe destacar la seguridad en el código. Durante el proyecto no nos centraremos en este aspecto, sin embargo, es importante saber que existe y que es necesario implementar medidas de seguridad a nivel de código.

Como bien sabemos 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.

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.

---

*g. 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, 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 realizamos pruebas de seguridad o pentesting en un servicio en producción, estamos llevando a cabo una estrategia de seguridad reactiva. En este proceso, buscamos 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 consiste en llevar a cabo acciones antes de lanzar o implementar el servicio en producción. Al utilizar mecanismos de seguridad proactiva, podemos prevenir y evitar vulnerabilidades en nuestros servicios que podrían ser explotadas por atacantes.

*h. Análisis del proceso de seguridad en microservicios*

**No se donde añadir este apartado**

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.

#### *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 la seguridad de aplicaciones:

Área de Seguridad	Recomendaciones
<b>Infraestructura de Kubernetes</b>	<ul style="list-style-type: none"> <li>• Implementar medidas de control de acceso y autenticación robustos.</li> <li>• Aplicar cifrado de datos en reposo y en tránsito.</li> <li>• Monitorear y registrar la actividad en la infraestructura.</li> <li>• Segmentar la red para reducir el riesgo y limitar la propagación de compromisos</li> </ul>
<b>Contenedores</b>	<ul style="list-style-type: none"> <li>▪ Seleccionar imágenes confiables provenientes de fuentes confiables.</li> <li>▪ Configurar los contenedores de manera segura con mínimo privilegio.</li> <li>▪ Proteger la comunicación entre los contenedores y con el exterior mediante la implementación de mecanismos de seguridad adecuados.</li> <li>▪ Gestionar y aplicar parches de seguridad y actualizaciones de manera regular.</li> </ul>
<b>Seguridad en los clústeres de Kubernetes</b>	<ul style="list-style-type: none"> <li>• Asegurar la configuración de los componentes del clúster.</li> <li>• Garantizar la seguridad de las aplicaciones que se ejecutan en el clúster.</li> <li>• Aplicar autenticación, autorización y cifrado de comunicación.</li> <li>• Implementar medidas de detección y respuesta a incidentes.</li> </ul>
<b>Seguridad en el código</b>	<ul style="list-style-type: none"> <li>▪ Adoptar prácticas de desarrollo seguro.</li> <li>▪ Utilizar herramientas de análisis estático seguro de código (SAST).</li> <li>▪ Gestionar adecuadamente las dependencias del código y mantenerlas actualizadas.</li> <li>▪ Fomentar la conciencia de seguridad y promover buenas prácticas de codificación</li> </ul>

<b>Imágenes Docker</b>	<ul style="list-style-type: none"><li>▪ Realizar un análisis exhaustivo de seguridad antes de la implementación.</li><li>▪ Utilizar herramientas automatizadas para identificar y corregir vulnerabilidades.</li><li>▪ Utilizar imágenes base confiables y mantenerlas actualizadas.</li><li>▪ Seguir las mejores prácticas en la construcción de imágenes Docker.</li><li>▪ Gestionar adecuadamente las dependencias de las imágenes.</li><li>▪ Implementar mecanismos de monitoreo y registro para detectar actividad sospechosa.</li></ul>
------------------------	---

**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 analizaremos las herramientas existentes para las diferentes áreas de seguridad y decidir cuáles emplear para proteger nuestro entorno.

## **5.4 Iteración 4 – Investigación de herramientas para proteger un entorno de Kubernetes**

### **5.4.1 Tareas US 4**

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

### **5.4.2 Desarrollo**

Una vez analizadas las distintas áreas a proteger durante el proyecto, nos enfocamos en las siguientes, ya que son las 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.

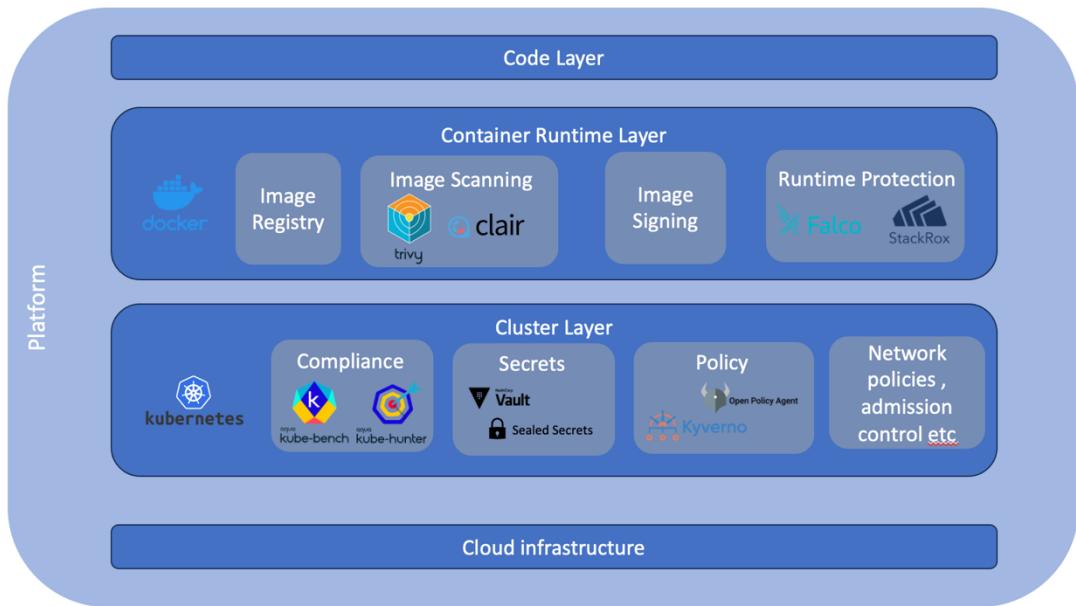
Se descubre que Kubernetes ofrece configuraciones específicas para la seguridad de algunas áreas como estándares de seguridad, admisiones y control de acceso, políticas de seguridad y secretos en Kubernetes. Por lo tanto, se concluye que Kubernetes no es capaz de cubrir todas las necesidades de seguridad.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

En nuestras búsquedas, se obtienen las siguientes herramientas. Ahora analicemos las principales herramientas más utilizadas<sup>20</sup>, incluyendo aquellas empleadas en una multinacional como NTTDATA. Y obtenemos la selección mostrada en la ilustración 10.

### **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 un diagrama que muestra las herramientas seleccionadas hasta el momento. Este diagrama representa una visión general de las opciones que hemos considerado. A medida que avanzamos en la investigación, continuaremos 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 10.** Selección de herramientas a analizar

---

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

### 5.5.1 Tareas US 5

01. Comparativas en función de las características
02. Instalación de las herramientas
03. Explicar funcionamiento 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 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, Secretos y Políticas. Obtendremos una tabla comparativa por sección.

Para cada uno de los ejemplos realizados en esta iteración se emplea la imagen nginx [25].

#### a. Compliance

**Kube-bench** [26] y **Kube-hunter** [27] 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.

- Kube-bench

A continuación, mostramos los resultados, así como la interfaz y funcionamiento de Kube-bench para comprender mejor la herramienta. Los pasos para la instalación se encuentran en la guía de instalación, en el anexo.

```
at * minikube
[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 11. Ejecución Kube-bench**

Kube-bench se ejecuta a través de la terminal y se obtienen los siguientes resultados.

```
== 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 12. Resultados de la herramienta Kube-bench**

En la ilustración 12, observamos los resultados obtenidos. Estos resultados se obtienen en forma de lista con un resumen por sección donde se visualiza los resultados de los test 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

A continuación, mostramos los resultados, así como la interfaz y funcionamiento de Kube-hunter para comprender mejor la herramienta. Los pasos para la instalación se encuentran en la guía de instalación, en el anexo.

```
at * minikube
2023-02-28 10:58:13,876 INFO kube_hunter.modules.report.collector Started hunting
2023-02-28 10:58:13,878 INFO kube_hunter.modules.report.collector Discovering Open Kubernetes Services
2023-02-28 10:58:13,903 INFO kube_hunter.modules.report.collector Found vulnerability "CAP_NET_RAW Enabled" in Local to Pod (kube-hunter-qd6tq)
2023-02-28 10:58:13,903 INFO kube_hunter.modules.report.collector Found vulnerability "Read access to pod's service account token" in Local to Pod (kube-hunter-qd6tq)
2023-02-28 10:58:13,905 INFO kube_hunter.modules.report.collector Found vulnerability "Access to pod's secrets" in Local to Pod (kube-hunter-qd6tq)
2023-02-28 10:58:19,140 INFO kube_hunter.modules.report.collector Found open service "Kubelet API" at 172.17.0.1:10250
2023-02-28 10:58:19,413 INFO kube_hunter.modules.report.collector Found open service "API Server" at 10.96.0.1:443
2023-02-28 10:58:19,452 INFO kube_hunter.modules.report.collector Found vulnerability "Access to API using service account token" in 10.96.0.1:443
2023-02-28 10:58:19,461 INFO kube_hunter.modules.report.collector Found vulnerability "K8s Version Disclosure" in 10.96.0.1:443
2023-02-28 10:58:29,299 INFO kube_hunter.modules.report.collector Found open service "Etcd" at 17.17.0.92:2379

Nodes
+-----+-----+
| TYPE | LOCATION |
+-----+-----+
| Node/Master | 172.17.0.92 |
| Node/Master | 172.17.0.1 |
| Node/Master | 10.96.0.1 |
```

**Ilustración 13.** Ejecución Kube-hunter

Kube-bench se ejecuta a través de la terminal y se obtienen los siguientes resultados.

Vulnerabilities					
For further information about a vulnerability, search its ID in: <a href="https://avd.aquasec.com/">https://avd.aquasec.com/</a>					
ID	LOCATION	MITRE CATEGORY	VULNERABILITY	DESCRIPTION	EVIDENCE
None	Local to Pod (kube-hunter-qd6tq)	Lateral Movement // ARP poisoning and IP spoofing	CAP_NET_RAW Enabled	CAP_NET_RAW is enabled by default for pods. If an attacker manages to compromise a pod, they could potentially take advantage of this capability to perform network attacks on other pods running on the same node	
KHV002	10.96.0.1:443	Initial Access // Exposed sensitive interfaces	K8s Version Disclosure	The kubernetes version could be obtained from the /version endpoint	v1.25.2
KHV005	10.96.0.1:443	Discovery // Access the K8s API Server	Access to API using service account token	The API Server port is accessible. Depending on your RBAC settings this could expose access to or control of your cluster.	b'{"kind": "APIServices", "versions": ["v1"], "serverAddressByClientCIDRs": [{"clientCIDR": "0.0.0.0/0", "serverAddress": "..."}]
None	Local to Pod (kube-hunter-qd6tq)	Credential Access // Access container service account	Access to pod's secrets	Accessing the pod's secrets within a compromised pod might disclose valuable data to a potential attacker	['/var/run/secrets/kubernetes.io/serviceaccount/namespace', '/var/run/secrets/kubernetes.io/serviceac...
KHV050	Local to Pod (kube-hunter-qd6tq)	Credential Access // Access container service account	Read access to pod's service account token	Accessing the pod service account token gives an attacker the option to use the server API	eyJhbGciOiJSUzI1NiSIntpZC16IjJ5SU1Ja1lqVXdyWUtU2RF0EZNWlMXjVXF0UmRVTFoT3pDWEZOUjA1f0.eyJhdWQiO...

**Ilustración 14.** Resultados de la herramienta Kube-hunter

En la ilustración 14, observamos los resultados en forma de tabla. En la tabla aparecen las vulnerabilidades detectadas con la categoría MITRE, la localización, la descripción y la evidencia. Esto resulta muy práctico a la hora de poder corregir las vulnerabilidades presentes en el clúster.

## b. Gestión de secretos

**Vault [28]** es una solución de gestión de secretos desarrollada por HashiCorp. Proporciona un repositorio centralizado para almacenar y controlar el acceso a contraseñas, claves API, certificados y otros datos sensibles. Vault ofrece un conjunto completo de características para la gestión de secretos, incluyendo generación, rotación, revocación y auditoría.

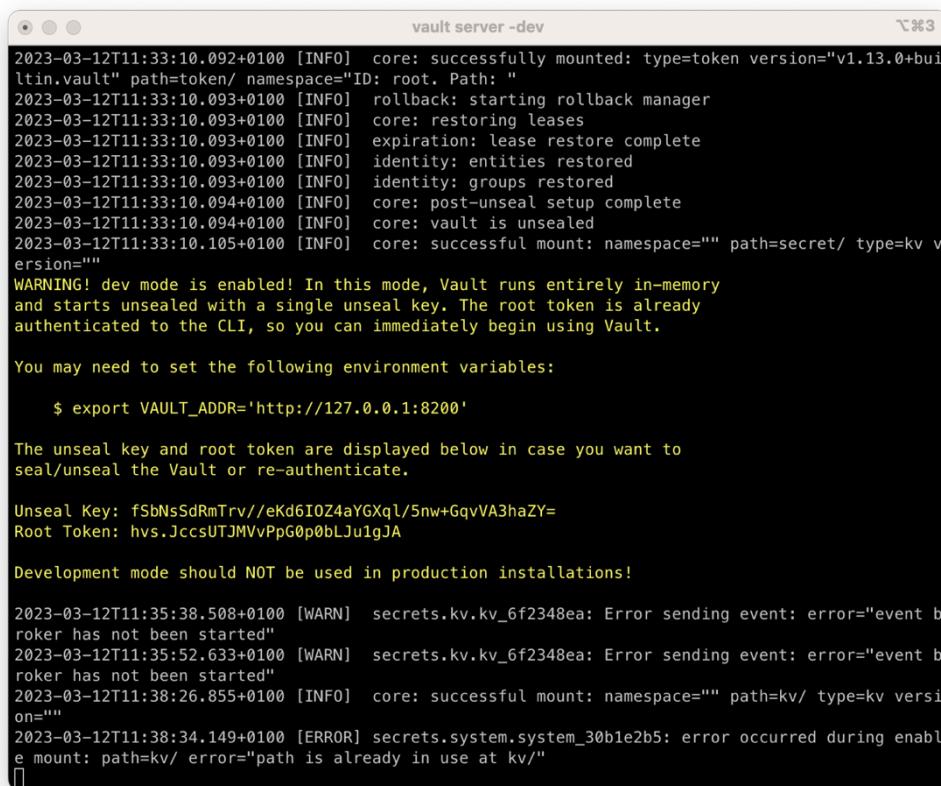
Por otro lado, **Sealed Secrets [29]** es una herramienta que permite el almacenamiento seguro y la gestión de secretos en Kubernetes. Sealed Secrets utiliza la criptografía asimétrica para cifrar los secretos y luego los almacena en el clúster de Kubernetes en forma de objetos sellados. Estos

objetos sellados pueden ser desencriptados 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**

A continuación, mostramos un ejemplo de secreto, así como la interfaz y funcionamiento de Vault para comprender mejor la herramienta. Los pasos para la instalación se encuentran en la guía de instalación, en el anexo.



```
vault server -dev
2023-03-12T11:33:10.092+0100 [INFO] core: successfully mounted: type=token version="v1.13.0+bui
ltin.vault" path=token/ namespace="ID: root. Path: "
2023-03-12T11:33:10.093+0100 [INFO] rollback: starting rollback manager
2023-03-12T11:33:10.093+0100 [INFO] core: restoring leases
2023-03-12T11:33:10.093+0100 [INFO] expiration: lease restore complete
2023-03-12T11:33:10.093+0100 [INFO] identity: entities restored
2023-03-12T11:33:10.093+0100 [INFO] identity: groups restored
2023-03-12T11:33:10.094+0100 [INFO] core: post-unseal setup complete
2023-03-12T11:33:10.094+0100 [INFO] core: vault is unsealed
2023-03-12T11:33:10.105+0100 [INFO] core: successful mount: namespace="" path=secret/ type=kv v
ersions=""
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: fSbNsSdRmTrv//eKd6I0Z4aYGXql/5nw+GqvVA3haZY=
Root Token: hvs.JccsUTJMvPpG0p0bLJu1gJA

Development mode should NOT be used in production installations!

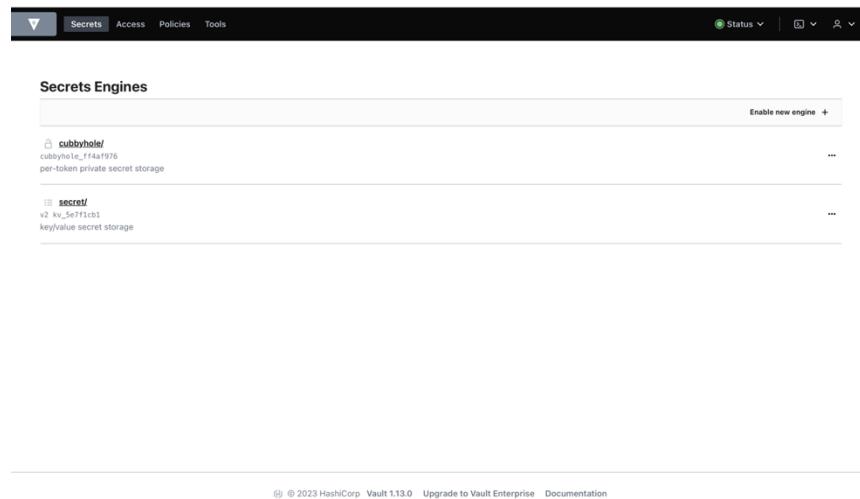
2023-03-12T11:35:38.508+0100 [WARN] secrets.kv.kv_6f2348ea: Error sending event: error="event b
roker has not been started"
2023-03-12T11:35:52.633+0100 [WARN] secrets.kv.kv_6f2348ea: Error sending event: error="event b
roker has not been started"
2023-03-12T11:38:26.855+0100 [INFO] core: successful mount: namespace="" path=kv/ type=kv versi
on=""
2023-03-12T11:38:34.149+0100 [ERROR] secrets.system.system_30b1e2b5: error occurred during enabl
e mount: path=kv/ error="path is already in use at kv/"


```

### **Ilustración 15. 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, se observa el Dashboard de Vault.



**Ilustración 16.** Dashboard Vault

A continuación mostramos un ejemplo de generación de secreto en Vault a través la línea de comandos.

```

andreavalvaromartin@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.

$ 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

$ vault kv put -mount=secret hello foo=world excited=yes
== Secret Path ==
secret/data/hello

===== Metadata =====
Key      Value
---      ---
created_time  2023-03-12T10:35:52.62982Z
custom_metadata  <nil>
deletion_time  n/a
destroyed      false
version        2

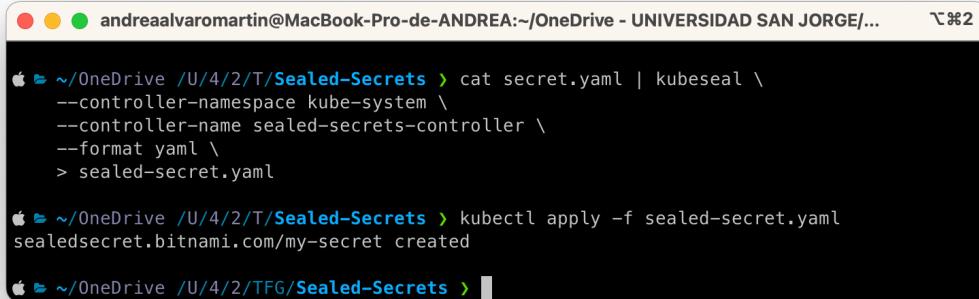
```

**Ilustración 17.** Dashboard Vault

- Sealed Secrets

A continuación, mostramos los resultados, así como la interfaz y funcionamiento de Sealed Secrets para comprender mejor la herramienta. Los pasos para la instalación se encuentran en la guía de instalación, en el anexo.

La gestión en Sealed Secrets es algo diferente. Se crean secretos de la siguiente forma.



The screenshot shows a terminal window with three tabs at the top: 'andreaalvaromartin@MacBook-Pro-de-ANDREA:~/OneDrive - UNIVERSIDAD SAN JORGE/...', 'Untitled 2', and 'Untitled 3'. The main pane displays the following command sequence:

```
~/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 18.** Ejecución Sealed Secrets

En la ilustración 18, se observa la creación de un secreto las características indicadas y se genera un archivo yaml, como se puede observar en el anexo 5 . Se pueden crear varios archivos sin embargo luego es necesario aplicarlo al entorno deseado para que se emplee correctamente.

#### c. 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 y gobernanza.

**Open Policy Agent [30]** 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 [31]** 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.

- Open Policy Agent

Las políticas de Kyverno se gestionan a través de la línea de comandos. A continuación, observamos un ejemplo de política aplicada. El file .yaml se encuentra en anexo 5. Los pasos para la instalación se encuentran en la guía de instalación, en anexo.

```

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.yaml
k8srequiredlabels.constraints,gatekeeper.sh/ns-must-have-gk created

andreaalvaromartin@MacBook-Pro-de-ANDREA:~/OneDrive - U/4/2_SEMESTER/TFG/OPA_GateKeeper> kubectl apply -f bad_NameSpace.yaml
Error from server (Forbidden): error when creating "bad_NameSpace.yaml": admission webhook "validation.gatekeeper.sh" denied the request: [ns-must-have-gk] you must provide
labels: {"gatekeeper"}

andreaalvaromartin@MacBook-Pro-de-ANDREA:~/OneDrive - U/4/2_SEMESTER/TFG/OPA_GateKeeper> kubectl apply -f good_NameSpace.yaml
namespace/mynamespace created

```

**Ilustración 19.** Aplicar política en OPA

- Kyverno

Las políticas en Kyverno se gestionan también a través de la línea de comandos. A continuación, observamos un ejemplo de política aplicada. El file. yaml se encuentra en anexo 5. Los pasos para la instalación se encuentran en la guía de instalación, en anexo.

```

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

```

**Ilustración 21.** Ejemplo aplicación de política

### 5.5.3 Resultados

#### a. Comparativas herramientas compliance

A continuación, se presenta una tabla comparativa de 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 CIS Kubernetes Benchmark.	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
Tipo de herramienta	Herramienta de benchmarking.	Herramienta de escaneo de vulnerabilidades. Se utiliza para buscar 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	Si. Se enfoca en las configuraciones de seguridad recomendadas por el NCSC de EE. UU.	Si. Busca posibles vulnerabilidades en la infraestructura del clúster de Kubernetes
Método de análisis	Compara las configuraciones del clúster con el benchmarking de CIS	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	Simple/ Medio	Simple
Facilidad de Instalación	+++++	++++ (Más pasos)
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 1.** 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.

*b. Comparativas herramientas sobre secretos: Vault y Sealed Secrets*

A continuación, se presenta una tabla 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
Integración con Kubernetes	<b>Específico para Kubernetes</b> y se integra con sus herramientas de gestión de secretos. Sealed Secrets es	<b>Puede integrarse con Kubernetes</b> para la gestión de secretos y autenticación de

	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
Dificultad de uso	+++++	+++ (Más potente)
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
Recommendación 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 2. Comparativa Sealed Secrets y Vault**

c. Comparativa Herramientas para 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. A continuación, se presenta una tabla 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 (Vía extensiones)
Verificación Manifiesto (Sigstore)	Si	No
Registro de Imagen	Si	Si (Vía extensiones)

Extensiones	No	Si (Version Beta)
Políticas como fuentes nativas	Si	Si
Exposición de Metricas	Si	Si
Open API esquema de validación (kubectl explain)	Si	No
Alta disponibilidad	Si	Si
API object lookup	Si	Si
CLI with test ability	Si	Si
Policy audit ability	Si	Si
Self-service reports	Si	No
Self-service exceptions	Si	No
Licencia	Apache 2.0	Apache 2.0
Dificultad	++++	+++
Programming required	Si	Si
Use outside Kubernetes	No	Si
Documentation maturity	●	■
Policy sample library	Si	Si
Number of community policy samples	46	265

**Tabla 3.** Comparativa Kyverno y Open Policy Agent

Con conclusión del análisis de Kyverno y Open Policy Agent – GateKeeper

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

**Tabla 4..** Comparativa OPA GateKeeper y Kyverno ventajas y desventajas

---

## **5.6 Iteración 6 – Análisis de las herramientas seleccionadas - Container Run time layer**

### **5.6.1 Tareas US 6**

01. Comparativas en función
02. Instalación de las herramientas
03. Explicar funcionamiento de las herramientas

### **5.6.2 Desarrollo**

### **5.6.3 Resultados**

---

## 5.7 Iteración 7 – Análisis Falco

### 5.7.1 Tareas US 7

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

### 5.7.2 Desarrollo

#### Herramientas para terceros

**Grandes proyectos => falco integrar alertas Cloud , AWS**

**Enlazar ergocd conectar => registrar que esta pasando**

**Falco rules mencionar => macro listas**

*Falco*

Falco es una solución de seguridad nativa de la nube basada en Kubernetes que analiza las llamadas al sistema y la actividad de las aplicaciones en tiempo real para detectar y prevenir problemas de seguridad. Emplea un conjunto de reglas para especificar el comportamiento anticipado y advierte sobre las violaciones, con acciones como detener los procesos ofensivos que son personalizables. Falco se basa en la tecnología Sysdig y puede identificar una amplia gama de riesgos en los sistemas Kubernetes, mejorando la seguridad y la defensa contra futuros ataques.

### 5.7.3 Resultados

---

## 5.8 Iteración 8 – Análisis StackRox

### 5.8.1 Tareas US 8

01. Análisis StackRox – funcionamiento herramienta
02. Instalar StackRox
03. Redactar Instalación StackRox – Manual de Usuario
04. Configurar StackRox
05. Buenas prácticas StackRox
06. Redactar Buenas prácticas Sysdig Secure – Manual Usuario

---

### *5.8.2 Desarrollo*

#### *StackRox*

StackRox es una tecnología de seguridad de contenedores que ofrece seguridad continua para la configuración de Kubernetes. A través de capacidades como la gestión de vulnerabilidades, la gestión de cumplimiento y el perfil de riesgos, identifica y mitiga las amenazas de seguridad. StackRox detecta y responde a los riesgos de seguridad utilizando el aprendizaje automático y se conecta con las tecnologías DevOps para una implementación y administración sin problemas.

### *5.8.3 Resultados*

---

## **5.9 Iteración 9 – Análisis Sysdig Secure**

### *5.9.1 Tareas US 9*

- 01. Análisis Sysdig – funcionamiento herramienta*
- 02. Instalar Sysdig*
- 03. Redactar Instalación Sysdig Secure – Manual Usuario*
- 03. Configurar Sysdig*
- 04. Buenas prácticas Sysdig Secure*

---

*05. Redactar Buenas prácticas Sysdig Secure*

*5.9.2 Desarrollo*

*5.9.3 Resultados*

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

*5.10.1 Tareas US 10*

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

*5.10.2 Desarrollo*

*5.10.3 Resultados*

Como resultado de este proyecto hemos realizado los siguientes productos:

**Tabla comparativa de las herramientas Stack Rox y Falco, (1 página. Explicación tabla )**

**Guía de Instalación de las herramientas y buenas prácticas recomendadas. (captura de la guía de instalación 1 página)**

## A. Estudio económico

### 8.1 Desglose de costes

#### a. Costes materiales

En cuanto a los gastos del proyecto, se pueden identificar distintas secciones según los materiales que se hayan utilizado durante su desarrollo, siendo distribuidos de la siguiente manera:

	Precio	Vida Útil	Tiempo usado	Coste Real
<b>Macbook pro 13"</b>	2240€	5 años	5 meses	2240€
<b>Periféricos</b>	100€	6 años	5 meses	100€

Pantalla 27"	120€	300000 horas	315h	10€
<b>Total</b>			2350€	

1. <https://www.ticpymes.es/tecnologia/noticias/1111246049504/vida-util-de-tecnologia-apple-puede-llegar-12-anos.1.html#:~:text=La%20vida%20%C3%AAtil%20de%20la,a%C3%B1os%20%7C%20N>  
[oticias%20%7C%20Tecnolog%C3%ADa%20%7C%20TicPymes&text=Si%20hay%20algo%20que%20ca.](https://www.ticpymes.es/tecnologia/noticias/1111246049504/vida-util-de-tecnologia-apple-puede-llegar-12-anos.1.html#:~:text=La%20vida%20%C3%AAtil%20de%20la,a%C3%B1os%20%7C%20N)
- 2.

<https://hardzone.es/2019/01/13/vida-util-monitor-antes-rompa/>

*b. Costes humanos*

En la tabla siguiente se presentan los costos 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
<b>Project manager</b>	20€/hour		
<b>Desarrollador</b>	18€/hour		
	Total	315 h	<b>Total</b>

*c. Costes de infraestructura*

**PREGUNTAR** ¿? Durante el proyecto, se necesitó un espacio de trabajo por ello se tiene en cuenta los siguientes factores (renta, agua, electricidad, Internet)

*d. Costes totales*

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

**HACER GRAFICO EXCEL O CANVA**

Costes Material	Costes Humanos	Coste de Infraestrutura	Total

## B. Resultados

"*¿Qué datos/productos se han obtenido al finalizar el proyecto?*"

*Resultados de simulaciones y procesos del Proyecto. Se deben reflejar las pruebas y test realizados, errores detectados...*

*Artefactos producidos.*

*Desviaciones de la metodología, acciones correctivas, riesgos eliminados o mitigados a lo largo de la vida del proyecto.*

---

En cuanto al resultado de la planificación del proyecto, en las siguientes ilustraciones podemos observar cómo se vio afectada la planificación inicial y cuál fue el resultado final:

La implementación transcurrió de manera normal, hubo algunos cambios en la planificación inicial, que se muestran en la Ilustración XX, pero no se encontró ningún obstáculo insuperable que paralizara el curso del proyecto.

Finalmente, el tiempo total invertido en este proyecto es el siguiente:

AÑADIR

Grafico 1 – Comparativa entre el timepo estimado y el tiempo real por iteration.

Figura. Diagrama de Gantt del trabajo real. – Planificaion total del tiempo dedicado

Por último, en este proyecto hemos implementado 10 historias de usuario, divididas en XX tareas y sus correspondientes pruebas de aceptación, lo que suma un total de XX pruebas en total, todas las cuales han sido superadas exitosamente.

En resumen, hemos logrado implementar con éxito las 10 historias de usuario, completando todas las tareas y superando satisfactoriamente las pruebas de aceptación asociadas.

## C. Conclusiones

FALTA :

"¿Lo realizado se corresponde con lo previsto?" En este apartado se debe reflejar el grado de cumplimiento del objetivo o de los objetivos planteados.

### a. Propuesta de mejora

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

- **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.
- **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.
- **Añadir en manual de usuario** la instalación de todas las herramientas necesarias para proteger Kubernetes a todos los niveles.
- Incluir **recomendaciones** de todas las herramientas de seguridad mencionadas.
- **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.

## D. Bibliografía

[1] Harris, Chandler. "Comparacion 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]

[1] [En linea]. Available: <https://kubernetes.io/docs/concepts/services-networking/network-policies/>

- [2] [En linea]. Available: <https://aprendearquitecturasoftware.wordpress.com/2018/09/27/arquitectura-spotify-microservicios/>
- [3] [En linea]. Available: <https://medium.com/@cuemby/grandes-empresas-que-usan-kubernetes-d73d56334243>
- [4] [En linea]. Available: <https://aws.amazon.com/es/microservices/>
- [5] [En línea]. Available: <https://www.atlassian.com/es/microservices/cloud-computing/containers-vs-vms>
- [6] [En linea]. Available: <https://www.docker.com>
- [7] [En línea]. Available: <https://kubernetes.io/es/>
- [8][En línea]. Available: [https://www.cncf.io/wp-content/uploads/2020/11/CNCF\\_Survey\\_Report\\_2020.pdf](https://www.cncf.io/wp-content/uploads/2020/11/CNCF_Survey_Report_2020.pdf)
- [9] [En linea]. Available: <https://www.datadoghq.com/container-report/>
- [10] [En linea]. Available: <https://www.redhat.com/es/resources/2021-global-tech-outlook-detail>
- [11] [En linea]. Available: <https://enlyft.com/tech/products/kubernetes>
- [12] [En linea]. Available: [https://www.redhat.com/es/topics/devops#:~:text=DevOps%20es%20un%20modo%20de,de%20los%20servicios%20de%20TI.](https://www.redhat.com/es/topics/devops#:~:text=DevOps%20es%20un%20modo%20de,de%20los%20servicios%20de%20TI)
- [13][En linea]. Available: [https://www.vmware.com/es/topics/glossary/content/devsecops.html#:~:text=<DevSecOps>%20\(abreviatura%20de%20«,diseñar%20aplicaciones%20sólidas%20y%20seguras.](https://www.vmware.com/es/topics/glossary/content/devsecops.html#:~:text=<DevSecOps>%20(abreviatura%20de%20«,diseñar%20aplicaciones%20sólidas%20y%20seguras.)
- [14] [EN LINEA]. Available: <https://www.atlassian.com/es/devops/what-is-devops/how-to-start-devops>
- [15] [En linea]. Available: <https://asana.com/es/resources/extreme-programming-xp>
- [16] [En linea]. Available: <https://trello.com>
- [17] [En linea]. Available: <https://www.microsoft.com/es-es/microsoft-365/excel>
- [18]
- [19]
- [20] [En línea o PDF]. Available: <https://www.redhat.com/rhdc/managed-files/cl-state-kubernetes-security-report-262667-202304-en.pdf>
- [21] [En linea]. Available: <https://kubernetes.io/docs/concepts/overview/>
- [22] [En linea]. Available: <https://kubernetes.io/docs/concepts/overview/components/>

- [23] [En linea]. Available: <https://kubernetes.io/docs/concepts/security/overview/>
- [24] [En linea]. Available: <https://aws.amazon.com/es/security/>
- [25][En linea]. Available: <https://www.nginx.com>
- [26] [En linea]. Available: <https://hub.docker.com/r/aquasec/kube-bench>
- [27] [En linea]. Available: <https://github.com/aquasecurity/kube-hunter>
- [28] [En linea]. Available: <https://developer.hashicorp.com/vault/docs/platform/k8s#>
- [29] [En linea]. Available: <https://github.com/bitnami-labs/sealed-secrets>
- [30] [En linea]. Available: <https://www.openpolicyagent.org>
- [31] [En linea]. Available: <https://kyverno.io>
- [32] [En linea]. Available:
- [33] [En linea]. Available:
- [34] [En linea]. Available:
- [35] [En linea]. Available:
- [36] [En linea]. Available:
- [37] [En linea]. Available:
- [38] [En linea]. Available:
- [39] [En linea]. Available:
- [40] [En linea]. Available:
- [41] [En linea]. Available:
- [42] [En linea]. Available:
- [43] [En linea]. Available:
- [2] [En linea]. Available: <https://docs.tigera.io/calico/3.25/getting-started/kubernetes/self-managed-onprem/onpremises>*

## E. Anexos

a. **Anexo 1: Propuesta**

**Nombre alumno:** \_\_\_\_\_

**Titulación:** \_\_\_\_\_

**Curso académico:** \_\_\_\_\_

#### **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.

#### **b. Anexo 2: Reuniones**

AÑADIR

**c. Anexo 3: Historias de Usuario**

ID: 0	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 01	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 02	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 03	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 04	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 05	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 06	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 07	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 08	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 09	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

ID: 10	Titulo:	
Descripción:		
Prioridad: Alta	Riesgo: Bajo	Esfuerzo: 30 h

**d. Anexo 4: Horas trabajadas**

AÑDIR GRAFICOS

**e. Anexo 5: Código YAML**

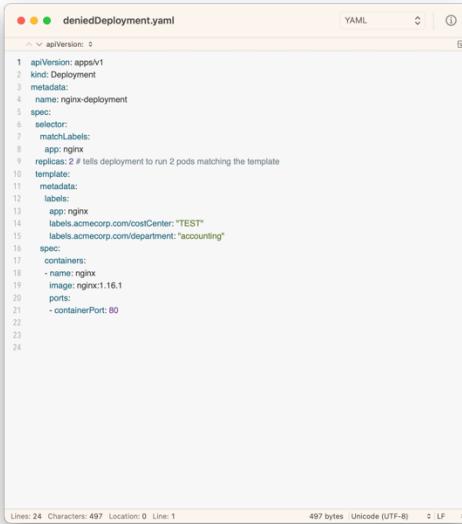
Sealed Secrets Ejemplo

Kyverno una por cada tipo que existe de politicas (Mutation....)

The screenshot shows a code editor window with the following details:

- Title Bar:** The title bar displays the file name "require\_runAsUser.yaml" and the word "YAML".
- Status Bar:** The bottom status bar indicates "Lines: 28 Characters: 623 Location: 0 Line: 1" and "823 bytes Unicode (UTF-8)".
- Code Content:** The main area contains the following YAML configuration:

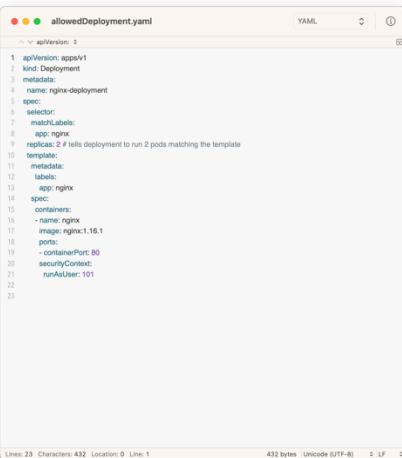
```
apiVersion: kymo.io/v1
kind: ClusterPolicy
metadata:
  name: require-runAsUser
spec:
  policies.kymo.io/initial: Require the runAsUser value to be Specified
  policies.kymo.io/category: Best Practices
  policies.kymo.io/severity: medium
  policies.kymo.io/object: Pod
  policies.kymo.io/description: >
    Requires Pods to specify runAsUser value within their containers which are not root.
    spec:
      validation/failureAction: enforce
      background: false
      rules:
        - name: check-userid
          match:
            resources:
              kinds:
                - Pod
            validate:
              message: >-
                The field spec.containers.*.securityContext.runAsUser must specified and greater than zero.
            pattern:
              spec:
                containers:
                  - securityContext:
                      runAsUser: '>0'
```



```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
        labels.acmecorp.com/costCenter: "TEST"
        labels.acmecorp.com/department: "accounting"
    spec:
      containers:
        - name: nginx
          image: nginx:1.16.1
          ports:
            - containerPort: 80
  
```

Lines: 24 Characters: 497 Location: 0 Line: 1 497 bytes Unicode (UTF-8) ⌂ LF ⌂

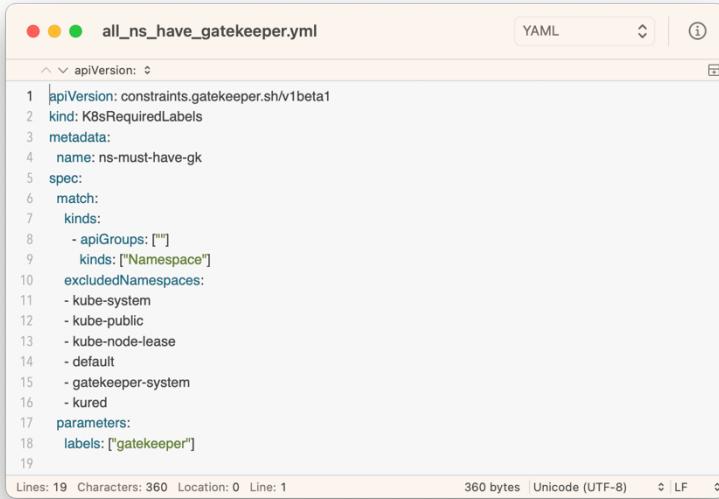


```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
        labels.acmecorp.com/costCenter: "TEST"
        labels.acmecorp.com/department: "accounting"
    spec:
      containers:
        - name: nginx
          image: nginx:1.16.1
          ports:
            - containerPort: 80
          securityContext:
            runAsUser: 101
  
```

Lines: 23 Characters: 432 Location: 0 Line: 1 432 bytes Unicode (UTF-8) ⌂ LF ⌂

## OPA - GateKeeper



The screenshot shows a code editor window with the following content:

```
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sRequiredLabels
metadata:
  name: ns-must-have-gk
spec:
  match:
    kinds:
      - "Namespace"
  excludedNamespaces:
    - kube-system
    - kube-public
    - kube-node-lease
    - default
    - gatekeeper-system
    - kured
  parameters:
    labels: ["gatekeeper"]
```

Details from the bottom of the editor:

Lines: 19 Characters: 360 Location: 0 Line: 1      360 bytes Unicode (UTF-8)      LF

**f. Anexo 6: Guia de instalación**