

Guía de Instalación

De herramientas para el Entorno de Kubernetes

En MacOS

Herramientas actuales para proteger su entorno

Tabla de contenido

1.	ANTES DE INSTALAR.....	3
2.	PREPARACIÓN A MINIKUBE.....	4
2.1	CONSIDERACIONES PREVIAS NECESARIAS.....	4
2.1.1	<i>Requisitos Docker.....</i>	<i>4</i>
2.1.2	<i>Requisitos Homebrew.....</i>	<i>4</i>
2.1.3	<i>Requisitos Kubectl.....</i>	<i>4</i>
2.2	INSTALACIÓN HERRAMIENTAS PREVIAS.....	5
2.2.1	<i>Instalación Docker.....</i>	<i>5</i>
2.2.2	<i>Instalación Homebrew.....</i>	<i>5</i>
2.2.3	<i>Instalación Kubectl.....</i>	<i>6</i>
2.2.4	<i>Instalación Helm.....</i>	<i>7</i>
3.	INSTALACIÓN MINIKUBE	8
3.1	REQUISITOS DEL SISTEMA PARA INSTALAR MINIKUBE	8
3.2	INSTALACIÓN MINIKUBE EN MACOS	8
3.3	EJEMPLO DE CREACIÓN DE SERVICIO.....	10
3.4	GESTIONAR EL CLÚSTER	10
3.5	BUENAS PRÁCTICAS EN MINIKUBE	11
4.	ANTES DE INSTALAR LAS HERRAMIENTAS.....	12
5.	INSTALACIÓN HERRAMIENTAS.....	14
5.1	INSTALACIÓN HERRAMIENTAS A NIVEL DE CLÚSTER	14
5.1.1	<i>A nivel de compliance.....</i>	<i>14</i>
5.1.2	<i>A nivel de gestión de secretos.....</i>	<i>21</i>
5.1.3	<i>A nivel de gestión de políticas.....</i>	<i>29</i>
5.2	INSTALACIÓN HERRAMIENTAS A NIVEL DE TIEMPO DE EJECUCIÓN EN CONTENEDORES	35
5.2.1	<i>A nivel de Escaneo de imágenes</i>	<i>35</i>
5.2.2	<i>A nivel de tiempo de ejecución.....</i>	<i>42</i>
6.	CASOS DE USO - TABLA RESUMEN	47

1. Antes de instalar

Proteger Kubernetes es de suma importancia debido a su papel fundamental en la gestión y orquestación de contenedores en entornos de producción. Aunque Kubernetes ofrece una plataforma flexible y escalable para ejecutar aplicaciones en contenedores, también plantea desafíos de seguridad que deben abordarse de manera adecuada. Es crucial instalar herramientas de protección específicas para el entorno de Kubernetes a fin de garantizar la seguridad y la integridad de las aplicaciones y los datos que se ejecutan en él.

El propósito principal de estas herramientas de protección es mitigar los riesgos de seguridad y fortalecer las defensas del clúster de Kubernetes. En esta guía de instalación, se proporcionarán pasos detallados para facilitar el uso del entorno y familiarizarse con las herramientas de protección.

Estas herramientas están diseñadas para identificar, prevenir y responder a posibles amenazas y vulnerabilidades que podrían comprometer la seguridad de los contenedores, los pods y los servicios en el entorno de Kubernetes. Al instalar estas herramientas, se busca establecer una capa adicional de seguridad que permita detectar y responder de manera eficiente a posibles ataques o incidentes de seguridad.

Aunque la guía se enfocará en el sistema operativo macOS, es importante tener en cuenta que las herramientas y los pasos pueden variar según los requisitos y las necesidades de seguridad de cada entorno de Kubernetes. Además, los valores de tiempo estimado y facilidad de instalación que se proporcionan son aproximados y se basan en un entorno de prueba. Por lo tanto, se recomienda adaptar la guía según las circunstancias particulares y mantenerse actualizado con las mejores prácticas de seguridad en Kubernetes.

2. Preparación a Minikube

Este capítulo describe los pasos necesarios para preparar el entorno para la instalación de Minikube en el sistema.

2.1 Consideraciones previas necesarias

Para proceder a la instalación de Minikube en macOS, es necesario tener previamente instaladas tres herramientas: Docker, Homebrew, kubectl y Helm. Estas herramientas son indispensables y se utilizarán a lo largo del proceso de instalación.

2.1.1 Requisitos Docker

A continuación, se muestran los requisitos necesarios para instalar Docker:

- El sistema macOS debe ser la versión 11 o posterior. Eso incluye Big Sur (11), Monterey (12) o Ventura (13). Recomendamos actualizar a la última versión de macOS.
- Al menos 4 GB de RAM.
- No se debe instalar VirtualBox antes de la versión 4.3.30, ya que no es compatible con Docker Desktop.

2.1.2 Requisitos Homebrew

A continuación, se muestran los requisitos necesarios para instalar Homebrew:

- Un CPU Intel de 64 bits o un CPU Apple Silicon
- macOS Big Sur (11) (o superior)
- Herramientas de línea de comandos (CLT) para Xcode (a través de `xcode-select --install` o desde <https://developer.apple.com/download/all/>) o Xcode
- Shell Bourne-again para la instalación (por ejemplo, bash)

2.1.3 Requisitos Kubectl

No hay requisitos específicos que sean de vital importancia. Sin embargo, es importante tener en cuenta lo siguiente: se recomienda utilizar una versión de kubectl que esté dentro de una diferencia de versión menor con respecto al clúster de Kubernetes. Por ejemplo, un cliente v1.27 puede comunicarse con las versiones v1.26, v1.27 y v1.28 del plano de control. Utilizar la última versión de kubectl ayuda a evitar problemas inesperados.

2.2 Instalación herramientas previas

2.2.1 Instalación Docker

Docker es una plataforma de contenedores ampliamente utilizada. Proporciona un entorno para crear, administrar y ejecutar contenedores. La versión de Docker para Mac se instala como una aplicación y tal cual vive dentro del directorio Aplicaciones, el instalador se encarga de crear vínculos simbólicos que permiten utilizar Docker dentro de la terminal de Mac.

Para instalar Docker en Mac:

1. Descargar Docker para Mac en: <https://www.docker.com/get-started/>
2. Seleccionar: Descarga para Mac – Intel Chip :

https://desktop.docker.com/mac/main/amd64/Docker.dmg?utm_source=docker&utm_medium=webreferral&utm_campaign=docs-driven-download-mac-amd64&gl=1*hgyf11*_ga*MTgwODYyMzk2MS4xNjgxMjA5ODY2*_ga_XJWPQMJYHQ*MTY4Njk5ODY1NC4xOC4xLjE2ODY5OTg3NTYuNTIuMC4w

3. Al finalizar la descarga ejecute el instalador, este abrirá un directorio con la aplicación.
4. Copie la aplicación dentro del directorio Applications de Mac.
5. Ejecute la aplicación Docker que copió dentro de Applications.
6. Aparecerá el icono de Docker dentro de la barra mostrando que se está inicializando Docker.

Mas información en la documentación oficial: <https://docs.docker.com/desktop/install/mac-install/>

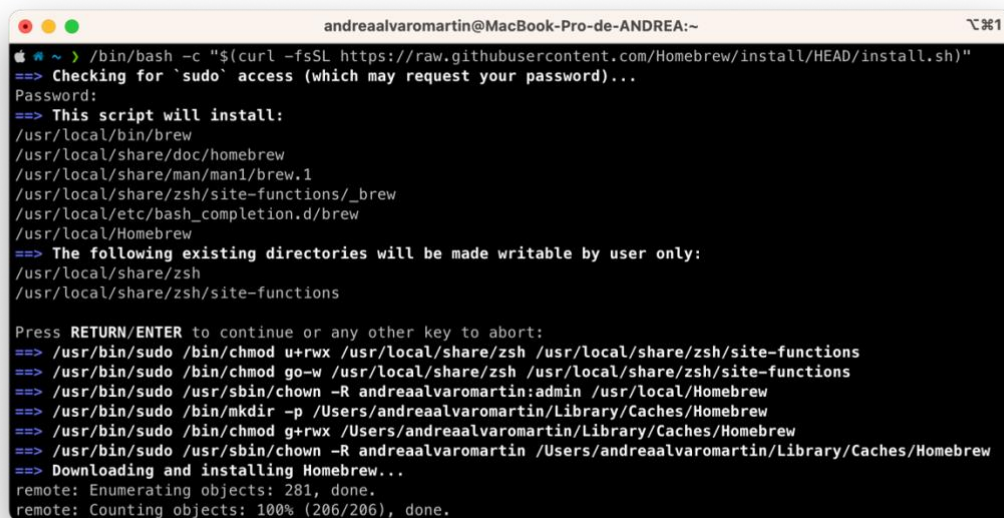
2.2.2 Instalación Homebrew

Homebrew es un administrador de paquetes popular para macOS que facilita la instalación y gestión de herramientas y software de código abierto en el sistema. Para instalar Homebrew:

1. Abrir una terminal
2. Ejecutar el siguiente comando:

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```





```
andreaalvaromartin@MacBook-Pro-de-ANDREA:~  
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"  
==> Checking for `sudo` access (which may request your password)...  
Password:  
==> This script will install:  
/usr/local/bin/brew  
/usr/local/share/doc/homebrew  
/usr/local/share/man/man1/brew.1  
/usr/local/share/zsh/site-functions/_brew  
/usr/local/etc/bash_completion.d/brew  
/usr/local/Homebrew  
==> The following existing directories will be made writable by user only:  
/usr/local/share/zsh  
/usr/local/share/zsh/site-functions  
Press RETURN/ENTER to continue or any other key to abort:  
==> /usr/bin/sudo /bin/chmod u+rw /usr/local/share/zsh /usr/local/share/zsh/site-functions  
==> /usr/bin/sudo /bin/chmod go-w /usr/local/share/zsh /usr/local/share/zsh/site-functions  
==> /usr/bin/sudo /usr/sbin/chown -R andreaalvaromartin:admin /usr/local/Homebrew  
==> /usr/bin/sudo /bin/mkdir -p /Users/andreaalvaromartin/Library/Caches/Homebrew  
==> /usr/bin/sudo /bin/chmod g+rw /Users/andreaalvaromartin/Library/Caches/Homebrew  
==> /usr/bin/sudo /usr/sbin/chown -R andreaalvaromartin /Users/andreaalvaromartin/Library/Caches/Homebrew  
==> Downloading and installing Homebrew...  
remote: Enumerating objects: 281, done.  
remote: Counting objects: 100% (206/206), done.
```

Ilustración 1. Instalación Homebrew

Siga las instrucciones en pantalla para completar la instalación de Homebrew.

Mas información en la documentación oficial: <https://docs.brew.sh/Installation>

2.2.3 Instalación Kubectl

Kubectl es una herramienta de línea de comandos que se utiliza para interactuar con clústeres de Kubernetes. Si estás en macOS y utilizas Homebrew como administrador de paquetes, se puede instalar kubectl de manera sencilla siguiendo los siguientes pasos en la terminal:

1. Ejecute el comando de instalación:

Opción 1:

```
$ brew install kubectl
```

Opción 2:

```
$ brew install kubernetes-cli
```

Con este comando, Homebrew se encargará de descargar e instalar la última versión de kubectl en el sistema. Una vez completada la instalación, podrás comenzar a utilizar kubectl para administrar y controlar los clústeres de Kubernetes.

2. Para asegurarse de que la versión que instaló se encuentra actualizada, ejecute el siguiente comando:

```
$ kubectl version --client
```



3. Para verificar que kubectl esté configurado correctamente ejecute el siguiente comando que permite obtener el estado del clúster:

```
$ kubectl cluster-info
```

Mas información en la documentación oficial:

<https://kubernetes.io/es/docs/tasks/tools/included/install-kubectl-macos/>

2.2.4 Instalación Helm

Helm es una herramienta de administración de paquetes y despliegue de aplicaciones en Kubernetes. Proporciona una forma fácil y eficiente de definir, instalar y actualizar aplicaciones complejas en un clúster de Kubernetes.

Helm utiliza un concepto llamado "chart", que es un paquete predefinido que contiene todos los recursos y configuraciones necesarios para desplegar una aplicación en Kubernetes. Un chart puede incluir archivos de configuración, plantillas, dependencias y cualquier otro recurso necesario para el despliegue de la aplicación.

Se instala Helm en el dispositivo:

```
$ brew install helm
```



3. Instalación Minikube

Una vez estén instaladas estas tres herramientas (Docker, Homebrew, kubectl y Helm), es momento para proceder con la instalación de Minikube.

3.1 Requisitos del sistema para instalar Minikube

Antes de instalar el software descrito en este manual, asegúrese de que su dispositivo reúne los siguientes requisitos:

- 2 CPUs o más
- 2GB de memoria RAM disponible
- 20GB de espacio en el disco
- Conexión a Internet
- Contenedor o gestor de máquinas virtuales, como: Docker, QEMU, Hyperkit, Hyper-V, KVM, Parallels, Podman, VirtualBox, or VMware Fusion/Workstation

3.2 Instalación Minikube en macOS

A continuación, se detallan los pasos a seguir para instalar Minikube en macOS:

PASO 1. Para instalar la última versión estable de Minikube en macOS x86-64 utilizando Homebrew, ejecute el siguiente comando:

```
$ brew install minikube
```

Si después de la instalación de Minikube a través de Homebrew el comando "which minikube" no funciona, es posible que debas eliminar los enlaces antiguos de Minikube y enlazar el nuevo archivo binario instalado. Puedes hacerlo siguiendo estos pasos:

1. Abra una terminal.
2. Ejecuta el siguiente comando para eliminar los enlaces antiguos:

```
$ brew unlink minikube
```

3. A continuación, enlaza el nuevo archivo binario ejecutando el siguiente comando:

```
$ brew link minikube
```

Con estos pasos, deberías poder utilizar el comando "minikube" correctamente en el entorno de macOS.

PASO 2. Iniciar el clúster. Para iniciar el clúster siga los siguientes pasos:

1. Abra Docker
2. Abra una terminal con acceso de administrador (pero sin iniciar sesión como root)
3. Ejecute:

```
$ minikube start
```

Minikube está listo para su uso.

```

andreaalvaromartin@MacBook-Pro-de-ANDREA:~
$ minikube start
minikube v1.30.1 en Darwin 13.1
Using the docker driver based on existing profile
Starting control plane node minikube in cluster minikube
Pulling base image ...
Restarting existing docker container for "minikube" ...
Preparando Kubernetes v1.26.3 en Docker 23.0.2...
Configurando CNI bridge CNI ...
Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
  Using image docker.io/kubernetesui/dashboard:v2.7.0
  Using image docker.io/kubernetesui/metrics-scrapers:v1.0.8
Some dashboard features require the metrics-server addon. To enable all features please run:

    minikube addons enable metrics-server

Complementos habilitados: storage-provisioner, dashboard
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
took 47s
  
```

Ilustración 2. Ejecución del entorno Minikube

Para obtener una mayor visibilidad del estado del clúster, Minikube incluye un Dashboard, que permite familiarizarse fácilmente con el nuevo entorno:

```
$ minikube dashboard
```

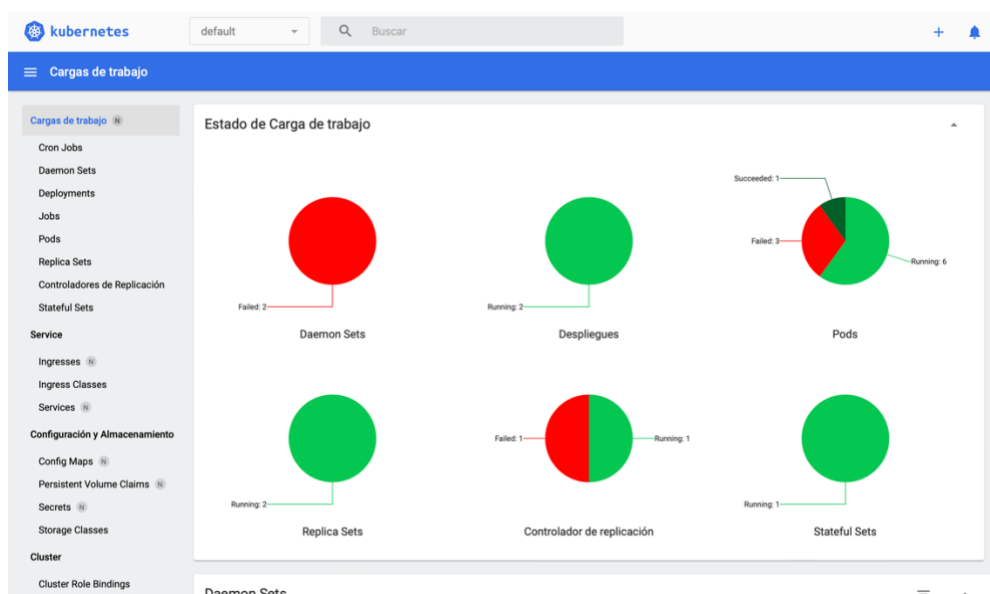


Ilustración 3. Dashboard de Minikube



3.3 Ejemplo de creación de servicio

A continuación, se muestra un ejemplo de despliegue en Kubernetes. El ejemplo completo se encuentra en la documentación oficial: <https://minikube.sigs.k8s.io/docs/start/>

PASO 1. Se crea una implementación de ejemplo y se expone en el puerto 8080.

```
$ kubectl create deployment hello-minikube --image=kicbase/echo-server:1.0
$ kubectl expose deployment hello-minikube --type=NodePort --port=8080
```

PASO 2. Compruebe que el despliegue esté funcionando ejecutando el siguiente comando:

```
$ kubectl get services hello-minikube
```

PASO 3. La forma más sencilla de acceder a este servicio es permitir que Minikube abra un navegador web.

```
$ minikube service hello-minikube
```

PASO 4. Alternativamente, usa kubectl para reenviar el puerto.

```
$ kubectl port-forward service/hello-minikube 7080:8080
```

PASO 5. Listo. La aplicación está disponible en <http://localhost:7080/>.

3.4 Gestionar el clúster

A continuación, se muestra una lista de comandos interesantes para la gestión de Minikube:

Comandos	Descripción
<code>kubectl get po -A</code>	Interactuar con el clúster, para acceder al clúster
<code>Kubectl get ns</code>	Muestra los namespaces existentes
<code>Kubectl get pods</code>	Muestra los pods existentes
<code>minikube pause</code>	Pausar Kubernetes sin afectar las aplicaciones desplegadas
<code>minikube unpause</code>	Reanudar una instancia pausada
<code>minikube stop</code>	Detener el clúster
<code>minikube addons list</code>	Explorar el catálogo de servicios de Kubernetes de fácil instalación
<code>minikube delete --all</code>	Eliminar todos los clústeres de Minikube

Tabla 1. Comandos útiles para gestionar Minikube

Para más información ir a la documentación oficial:

<https://minikube.sigs.k8s.io/docs/start/>



3.5 Buenas prácticas en Minikube

Aquí tienes una tabla que muestra algunas buenas prácticas al trabajar con Minikube:

Buena práctica	Descripción
Actualiza Minikube regularmente	Mantén la instalación de Minikube actualizada con la última versión para aprovechar las mejoras y correcciones.
Configura recursos adecuados	Asegurarse de tener asignados suficientes recursos de CPU, memoria y almacenamiento a el clúster Minikube.
Utiliza perfiles de Minikube	Los perfiles de Minikube permiten configurar diferentes opciones según los requisitos de los proyectos.
Habilita addons útiles	Minikube ofrece varios addons útiles como Dashboard, Ingress Controller, Metrics Server, entre otros.
Crea y utiliza imágenes Docker eficientes	Optimiza el tamaño de los imágenes Docker para reducir el consumo de recursos y mejorar la eficiencia. Además de instalar imágenes de fuentes fiables como las librerías oficiales: https://hub.docker.com
Utiliza volúmenes persistentes	Emplea volúmenes persistentes para almacenar datos de manera segura y persistente en las aplicaciones.
Practica el uso de Secrets y ConfigMaps	Utiliza Secrets para manejar información sensible y ConfigMaps para configurar variables de entorno.
Realiza pruebas y validaciones en entorno local	Antes de implementar en un entorno de producción, realiza pruebas y validaciones exhaustivas en Minikube.
Limpia y reinicia el clúster regularmente	Reinicia el clúster de Minikube y limpia recursos innecesarios para mantenerlo en un estado saludable.
Documenta y comparte las configuraciones	Documenta las configuraciones y comparte los archivos de configuración de Minikube con el equipo del proyecto.

Tabla 2. Buenas prácticas para gestionar Minikube

Se recuerda que estas son solo algunas de las buenas prácticas comunes, y es importante adaptarlas según las necesidades y requisitos específicos.



4. Antes de instalar las herramientas

La seguridad del entorno es de vital importancia. Por ello es necesario proteger las siguientes características:

- **Detección de vulnerabilidades:** Permiten escanear y analizar el entorno de Kubernetes en busca de posibles vulnerabilidades en los contenedores, las imágenes, las configuraciones o los componentes del clúster. Esto ayuda a identificar y corregir las debilidades de seguridad antes de que sean explotadas por atacantes.
- **Control de acceso y autenticación:** Proporcionan mecanismos sólidos para autenticar y autorizar el acceso a los recursos de Kubernetes. Esto incluye la gestión de usuarios, roles, permisos y políticas de seguridad para garantizar que solo los usuarios autorizados puedan interactuar con el clúster.
- **Monitoreo y registro de actividad:** Permiten supervisar y registrar la actividad en tiempo real dentro del entorno de Kubernetes. Esto ayuda a detectar y responder rápidamente a eventos sospechosos, identificar comportamientos anormales y realizar investigaciones forenses en caso de incidentes de seguridad.
- **Protección contra amenazas y ataques:** Ofrecen mecanismos de detección y prevención de intrusiones para proteger el entorno de Kubernetes contra ataques maliciosos, como intentos de ejecución remota de código, inyección de comandos, denegación de servicio, escalado no autorizado y otros tipos de amenazas.

A continuación, se muestra una imagen que ilustra los niveles clave a proteger, junto con las herramientas respectivas que se abordarán en esta guía de instalación. Para cada nivel, se proporcionará información sobre cómo instalar las herramientas necesarias y se compartirán una serie de mejores prácticas para su uso óptimo.

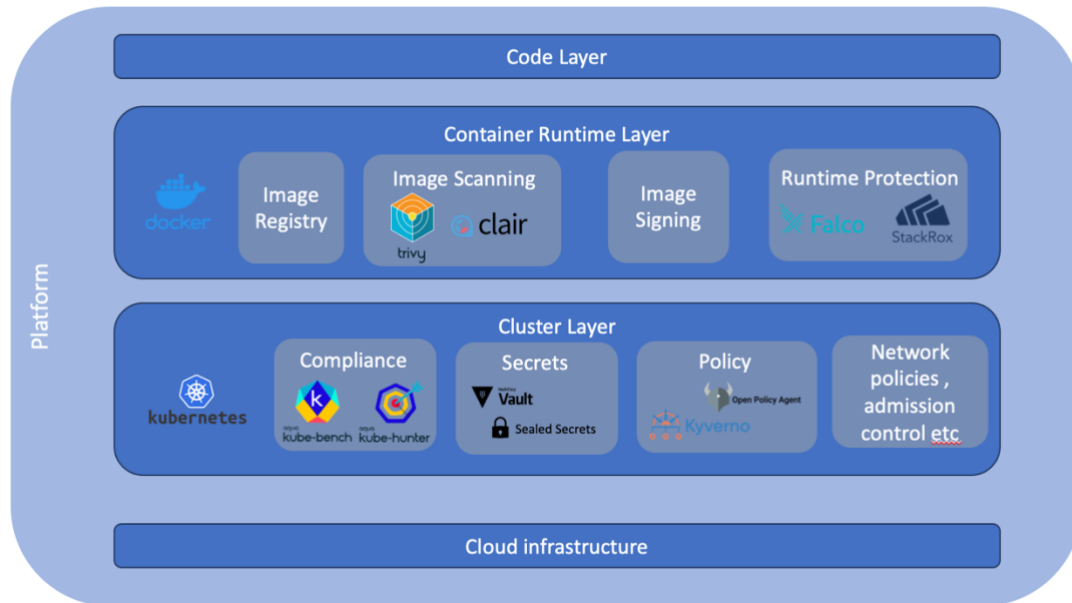


Ilustración 4. Diagrama por niveles y herramientas

Los usuarios se benefician de las siguientes maneras al utilizar estas herramientas:

- **Mayor seguridad:** Al instalar estas herramientas, los usuarios fortalecen la seguridad de sus entornos de Kubernetes y reducen la superficie de ataque potencial.
- **Detección temprana de amenazas:** Estas herramientas permiten identificar y abordar rápidamente las vulnerabilidades y los ataques, lo que ayuda a prevenir posibles violaciones de seguridad y minimiza el impacto en las aplicaciones y los datos.
- **Cumplimiento normativo:** Al implementar las herramientas de protección adecuadas, los usuarios pueden cumplir con los requisitos de seguridad y privacidad establecidos por las regulaciones y estándares industriales.
- **Mejor visibilidad y control:** Estas herramientas proporcionan una mayor visibilidad y control sobre el entorno de Kubernetes, lo que permite una administración más eficiente de la seguridad y una respuesta más rápida ante posibles incidentes.



5. Instalación herramientas

A continuación, se presentan los pasos de instalación para cada una de las herramientas.



5.1 Instalación herramientas a nivel de clúster

5.1.1 A nivel de compliance

5.1.1.1 Kube-bench

Kube-Bench es una herramienta de código abierto diseñada para evaluar la seguridad de los clústeres de Kubernetes. Realiza una serie de pruebas y verifica si se están siguiendo las mejores prácticas de seguridad recomendadas para los componentes principales de Kubernetes. Kube-Bench proporciona pautas claras y resultados detallados sobre posibles vulnerabilidades y configuraciones incorrectas que podrían afectar la seguridad del clúster. Esta herramienta es ampliamente utilizada para evaluar y mejorar la postura de seguridad en entornos de Kubernetes, ayudando a los administradores a identificar y abordar posibles problemas de seguridad antes de que sean explotados.

a. Instalación Kube-bench en Minikube

 10 min  Fácil

A continuación, se muestran los pasos de instalación:

PASO 1. Instala el archivo binario de Kube-bench para macOS utilizando los comandos que se muestran a continuación. Es importante tener en cuenta que podría haber versiones más recientes disponibles.

```
$ curl -L https://github.com/aquasecurity/kube-bench/releases/download/v0.6.2/kube-bench_0.6.2_linux_amd64.tar.gz -o kube-bench_0.6.2_linux_amd64.tar.gz  
$ tar -xvf kube-bench_0.6.2_linux_amd64.tar.gz
```

PASO 2. Ahora ya es posible ejecutar kube-bench con el siguiente comando:

```
$ kube-bench
```

PASO 3. Si has descargado manualmente el archivo binario de Kube-bench (usando el comando curl anterior), debes especificar la ubicación del directorio y archivo de configuración. Por ejemplo:

```
$ ./kube-bench --config-dir pwd/cfg --config pwd/cfg/config.yaml
```

En este caso, `pwd` se refiere al directorio actual donde se encuentra el archivo binario de kube-bench. Es necesario asegurarse de ajustar las rutas del directorio y archivo de configuración según corresponda a la configuración y ubicación actual.

PASO 4. Ejecutar Kube-bench en un clúster de Kubernetes:

Otro método para ejecutar Kube-bench es desplegándolo como un pod de trabajo de Kubernetes. Este método es especialmente útil para ejecutar las pruebas CIS en clústeres de Kubernetes gestionados donde no se tiene acceso de root al plano de control o a los nodos de trabajo.

1. Ejecuta el siguiente comando para desplegar Kube-bench utilizando el archivo `job.yaml` proporcionado por aquasecurity en GitHub:

```
$ kubectl apply -f https://raw.githubusercontent.com/aquasecurity/kube-bench/main/job.yaml
```

Si deseas modificar el archivo YAML, puedes descargarlo a un archivo local y luego aplicarlo mediante los siguientes comandos:

```
$ curl https://raw.githubusercontent.com/aquasecurity/kube-bench/main/job.yaml > job.yaml
```

```
$ kubectl apply -f job.yaml
```

Después de aplicar el archivo YAML, el informe de kube-bench estará disponible en los registros del pod.

2. Primero, listar los pods para obtener el nombre del pod:

```
$ kubectl get pods
```

```

$ kubectl apply -f job.yaml
job.batch/kube-bench created

$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
busybox       1/1     Running   4 (14h ago)  8d
kube-bench-h2zll 0/1     ContainerCreating 0          8s
nginx-76d6c9b8c-wkzkk 1/1     Running   4 (14h ago)  8d

$ kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
busybox       1/1     Running   4 (14h ago)  8d
kube-bench-h2zll 0/1     Completed 0          23s
nginx-76d6c9b8c-wkzkk 1/1     Running   4 (14h ago)  8d

```

Ilustración 5. Ejecución de Kube-bench

3. A continuación, se utiliza el nombre del pod para obtener los registros. Reemplaza "kube-bench-h2zll" con el nombre del pod:

```
$ kubectl logs kube-bench-h2zll
```



```

$ kubectl logs kube-bench-4j2bs
[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)
[PASS] 1.1.7 Ensure that the etcd pod specification file permissions are set to 600 or more restrictive (Automated)
[PASS] 1.1.8 Ensure that the etcd pod specification file ownership is set to root:root (Automated)
[WARN] 1.1.9 Ensure that the Container Network Interface file permissions are set to 600 or more restrictive (Manual)
[WARN] 1.1.10 Ensure that the Container Network Interface file ownership is set to root:root (Manual)
[FAIL] 1.1.11 Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)
[FAIL] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)
[PASS] 1.1.13 Ensure that the admin.conf file permissions are set to 600 or more restrictive (Automated)
[PASS] 1.1.14 Ensure that the scheduler.conf file permissions are set to 600 or more restrictive (Automated)
[PASS] 1.1.15 Ensure that the controller-manager.conf file permissions are set to 600 or more restrictive (Automated)
[PASS] 1.1.16 Ensure that the controller-manager.conf file ownership is set to root:root (Automated)
[FAIL] 1.1.17 Ensure that the Kubernetes PKI directory and file ownership is set to root:root (Automated)
[WARN] 1.1.18 Ensure that the Kubernetes PKI certificate file permissions are set to 600 or more restrictive (Manual)
[WARN] 1.1.19 Ensure that the Kubernetes PKI key file permissions are set to 600 (Manual)
[INFO] 1.2 API Server
[WARN] 1.2.1 Ensure that the --anonymous-auth argument is set to false (Manual)
[PASS] 1.2.2 Ensure that the --token-auth-file parameter is not set (Automated)
[PASS] 1.2.3 Ensure that the --deny-service-externalIPs is not set (Automated)
[PASS] 1.2.4 Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Automated)
[FAIL] 1.2.5 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)
[PASS] 1.2.6 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)
[PASS] 1.2.7 Ensure that the --authorization-mode argument includes Node (Automated)
[PASS] 1.2.8 Ensure that the --authorization-mode argument includes RBAC (Automated)
[WARN] 1.2.9 Ensure that the admission control plugin EventRelist is set (Manual)
[PASS] 1.2.10 Ensure that the admission control plugin AlwaysAdmit is not set (Automated)
[WARN] 1.2.11 Ensure that the admission control plugin AlwaysPullImages is set (Manual)
[PASS] 1.2.12 Ensure that the admission control plugin SecurityContextDeny is set if PodSecurityPolicy is not used (Manual)
[PASS] 1.2.13 Ensure that the admission control plugin ServiceAccount is set (Automated)
[PASS] 1.2.14 Ensure that the admission control plugin NamespaceLifecycle is set (Automated)
[PASS] 1.2.15 Ensure that the admission control plugin NodeRestriction is set (Automated)
[PASS] 1.2.16 Ensure that the --secure-port argument is not set to 0 (Automated)
[FAIL] 1.2.17 Ensure that the --profiling argument is set to false (Automated)
[FAIL] 1.2.18 Ensure that the --audit-log-path argument is set (Automated)
[FAIL] 1.2.19 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)
[FAIL] 1.2.20 Ensure that the --audit-log-maxsize argument is set to 10 or as appropriate (Automated)
[WARN] 1.2.21 Ensure that the --request-timeout argument is set as appropriate (Manual)
[PASS] 1.2.22 Ensure that the --service-account-key-file argument is set to true (Automated)
[PASS] 1.2.23 Ensure that the --service-account-key-file argument is set as appropriate (Automated)
[PASS] 1.2.24 Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Automated)

```

Ilustración 6. Logs de Kube-bench

Se obtiene los siguientes resultados del escaneo una vez ejecutado:

```

== 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 7. Resultados de Kube-bench

También puedes exportar los registros de kube-bench a un archivo:

```
$ kubectl logs kube-bench-4j2bs > kube-bench.report
```

Así tendrás el informe de kube-bench disponible en el archivo "kube-bench.report".

Más información en el GitHub de colaboración o la documentación oficial:

<https://github.com/aquasecurity/kube-bench/blob/main/docs/installation.md>

<https://aquasecurity.github.io/kube-bench/dev/installation/>



b. Buenas prácticas Kube-bench

A continuación, se resumen algunas buenas prácticas al utilizar Kube-bench en minikube:

Buenas prácticas	Descripción
Ejecutar Kube-bench regularmente	Ejecutar Kube-bench de forma periódica para evaluar el cumplimiento de las mejores prácticas de seguridad en el clúster minikube.
Analizar los resultados	Analizar los resultados generados por Kube-bench y comprender las áreas de mejora y los posibles riesgos de seguridad en el clúster minikube.
Corregir las vulnerabilidades	Tomar medidas para corregir las vulnerabilidades y los hallazgos de seguridad identificados por Kube-bench en el clúster minikube. Sigue las recomendaciones y las mejores prácticas proporcionadas para remediar los problemas.
Mantener actualizadas las versiones de Kubernetes y Kube-bench	Asegurarse de utilizar las versiones más recientes tanto de Kubernetes como de Kube-bench para aprovechar las últimas mejoras de seguridad y correcciones de errores.
Personalizar las configuraciones según las necesidades	Ajustar las configuraciones y los perfiles de evaluación de Kube-bench según las necesidades y requisitos específicos del clúster minikube.

Tabla 3. Buenas prácticas de Kube-bench

Kube-bench es una herramienta de evaluación de seguridad y no soluciona automáticamente los problemas identificados. Se deben tomar las medidas necesarias para corregir las vulnerabilidades y garantizar la seguridad del clúster minikube.



5.1.1.2 Kube-hunter

Kube-hunter es una herramienta de código abierto utilizada para detectar y explorar posibles vulnerabilidades en los clústeres de Kubernetes. Su objetivo es ayudar a los administradores de clústeres a identificar puntos débiles en la configuración y la seguridad de Kubernetes. Kube-hunter realiza exploraciones automáticas y pruebas de penetración en busca de posibles riesgos y exposiciones en el clúster. Proporciona información detallada sobre las vulnerabilidades encontradas, junto con recomendaciones para mitigar los riesgos. Esta herramienta es ampliamente utilizada para evaluar la seguridad de los clústeres de Kubernetes y mejorar su postura de seguridad.

a. Instalación Kube-hunter

 10 min  Fácil

A continuación, se muestran los pasos de instalación:

PASO 1. Dirigirse al nodo maestro de Kubernetes y ejecutar el siguiente comando:

```
$ kubectl create -f https://raw.githubusercontent.com/aquasecurity/kube-hunter/master/job.yaml
```

```
Apple ~ > kubectl create -f https://raw.githubusercontent.com/aquasecurity/kube-hunter/master/job.yaml
job.batch/kube-hunter created
```

Ilustración 8. Ejecución de la creación del nodo para Kube-Hunter

PASO 2. Espere hasta que se cree. Selecciona el nombre del pod a partir del resultado siguiente.

```
Apple ~ > kubectl get all                                     at * minikube
NAME                                READY    STATUS    RESTARTS   AGE
pod/busybox                         1/1     Running   4 (14h ago) 8d
pod/kube-bench-h2zll                0/1     Completed 0           5m42s
pod/kube-hunter-qd6tq               1/1     Running   0           16s
pod/nginx-76d6c9b8c-wkzkk          1/1     Running   4 (14h ago) 8d
```

Ilustración 9. Pods de Minikube

PASO 3. Compruebe los resultados de los logs del pods de Kube-hunter. Con el siguiente comando:

```
$ kubectl logs kube-hunter-qd6tq
```

Se obtienen los siguientes resultados en forma de tabla:



Ilustración 10. Resultados Kube-Hunter

Ilustración 11. Resultados Kube-Hunter 2

<https://aquasecurity.github.io/kube-hunter/>

b. Buenas prácticas Kube-Hunter

Aquí tienes algunas buenas prácticas al utilizar Kube-hunter en minikube

Buenas prácticas	Descripción
Ejecutar Kube-hunter regularmente	Ejecuta Kube-hunter de forma periódica para detectar posibles vulnerabilidades y debilidades en el clúster minikube.
Analizar los resultados	Analizar los resultados generados por Kube-hunter y comprende las áreas de riesgo y las vulnerabilidades identificadas en el clúster minikube.
Corregir las vulnerabilidades	Tomar medidas para corregir las vulnerabilidades y debilidades identificadas por Kube-hunter en el clúster minikube. Seguir las recomendaciones y las mejores prácticas proporcionadas para remediar los problemas.
Mantener actualizada la versión de Kube-hunter	Asegurarse de utilizar la versión más reciente de Kube-hunter para aprovechar las últimas mejoras de seguridad y correcciones de errores.
Implementar controles de seguridad adicionales	Utilizar los resultados de Kube-hunter como base para implementar controles de seguridad adicionales en el clúster minikube, como políticas de red, configuraciones de autenticación y autorización, y otras prácticas recomendadas de seguridad de Kubernetes.
Monitorear y auditar el clúster minikube	Implementar soluciones de monitoreo y auditoría para identificar y mitigar las amenazas de seguridad en tiempo real en el clúster minikube. Esto ayudará a detectar cualquier actividad sospechosa o violaciones de seguridad.

Tabla 4. Buenas prácticas para la gestión de Kube-Hunter

Es necesario mencionar que Kube-Hunter es una herramienta de detección de amenazas y no soluciona automáticamente los problemas identificados. Se deben tomar las medidas necesarias para corregir las vulnerabilidades y garantizar la seguridad en minikube.



5.1.2 A nivel de gestión de secretos

5.1.2.1 Vault

Vault es una herramienta de código abierto desarrollada por HashiCorp que se utiliza para gestionar de forma segura secretos, claves y otros datos sensibles en entornos de aplicaciones. Proporciona un almacenamiento centralizado y cifrado para secretos, lo que permite un mejor control y administración de la seguridad de los datos confidenciales. Vault ofrece funciones como generación de contraseñas, almacenamiento de tokens de autenticación, gestión de claves criptográficas y cifrado de datos. Además, Vault cuenta con características avanzadas de control de acceso y auditoría para garantizar la seguridad y el cumplimiento normativo. Es ampliamente utilizado en entornos de desarrollo y producción para proteger y gestionar secretos de manera segura.

a. Instalación Vault en Minikube



30 min



Intermedio

A continuación, se muestran los pasos a seguir para instalar Vault en Minikube.

Requisitos

- Interfaz de línea de comandos (CLI) de Kubernetes
- CLI de Helm
- Minikube
- Los gráficos de Helm de Vault

Instalación Vault

A continuación, se muestran los pasos de instalación de Vault.

PASO 1. Ejecutar el clúster Minikube

```
$ minikube start
```

PASO 2. Verificar el estado del clúster de Minikube

```
$ minikube status
```

PASO 3. Crea un namespace para instalar Vault y comprobar que se ha creado correctamente:

```
$ kubectl create ns vault
```

```
$ kubectl get ns
```

PASO 4. Vault gestiona los secretos que se escriben en estos volúmenes montables. Para proporcionar estos secretos, se requiere un único servidor de Vault. Para esta demostración, Vault puede ejecutarse en modo de desarrollo para manejar automáticamente la inicialización,



desbloqueo y configuración de un motor de secretos KV. Agrega el repositorio de Helm de HashiCorp:

```
$ helm repo add hashicorp https://helm.releases.hashicorp.com
```

PASO 5. Actualiza todos los repositorios para asegurar que Helm está en la última versión.

```
$ helm repo update
```

PASO 6. Para verificar, buscar en los repositorios los gráficos de Helm para Vault.

```
$ helm search repo hashicorp/vault
```

PASO 7. Instalar la última versión del gráfico de Helm de Vault con almacenamiento integrado (Integrated Storage):

```
$ cat > helm-vault-raft-values.yml <<EOF
```

```
server:
  affinity: ""
  ha:
    enabled: true
    raft:
      enabled: true
EOF
```

PASO 8. Instalar Vault Helm Chart.

```
$ helm install vault hashicorp/vault --values helm-vault-raft-values.yml
```

Esto crea tres instancias del servidor Vault con un backend de almacenamiento integrado (Raft).

PASO 9. Muestra todos los pods dentro del espacio de nombres (namespace) predeterminado.

```
$ kubectl get pods
```

PASO 10. Inicializa vault-0 con una parte de clave (key share) y un umbral de clave (key threshold) de uno.

```
$ kubectl exec vault-0 -- vault operator init \
  -key-shares=1 \
  -key-threshold=1 \
  -format=json > cluster-keys.json
$ cat cluster-keys.json
```

PASO 11. Crea una variable llamada VAULT_UNSEAL_KEY para capturar la clave de desbloqueo de Vault.

```
$ VAULT_UNSEAL_KEY=$(jq -r ".unseal_keys_b64[]" cluster-keys.json)
```

Después de la inicialización, Vault está configurado para saber dónde y cómo acceder al almacenamiento, pero no sabe cómo descifrar ninguno de los datos. El desbloqueo es el proceso



de construir la clave raíz necesaria para leer la clave de descifrado y descifrar los datos, lo que permite el acceso a Vault.

PASO 12. Desbloquea Vault que se está ejecutando en el pod vault-0.

```
$ kubectl exec vault-0 -- vault operator unseal $VAULT_UNSEAL_KEY
```

El servidor de Vault ha sido inicializado y desbloqueado.

PASO 13. Une el pod de vault-1 y vault-2 al clúster Raft.

```
$ kubectl exec -ti vault-1 -- vault operator raft join http://vault-0.vault-internal:8200
```

```
$ kubectl exec -ti vault-2 -- vault operator raft join http://vault-0.vault-internal:8200
```

PASO 14. Utiliza la clave de desbloqueo de arriba para desbloquear vault-1 y Vault-2

```
$ kubectl exec -ti vault-1 -- vault operator unseal $VAULT_UNSEAL_KEY
```

```
$ kubectl exec -ti vault-2 -- vault operator unseal $VAULT_UNSEAL_KEY
```

Después de este proceso todos los pods de Vault están ejecutándose (1/1 ready)

```
$ kubectl get po -n vault
```

PASO 15. El servicio de Vault es de tipo ClusterIP, lo que significa que podemos acceder a la consola de Vault desde un navegador. Para acceder a ella, necesitamos utilizar el comando de port-forward.

```
$ kubectl port-forward service/vault -n vault 8200:8200
```

PASO 16. Conecta <http://localhost:8200> en el navegador e ingresa el token raíz. Una vez se inicia sesión se obtiene el siguiente dashboard.

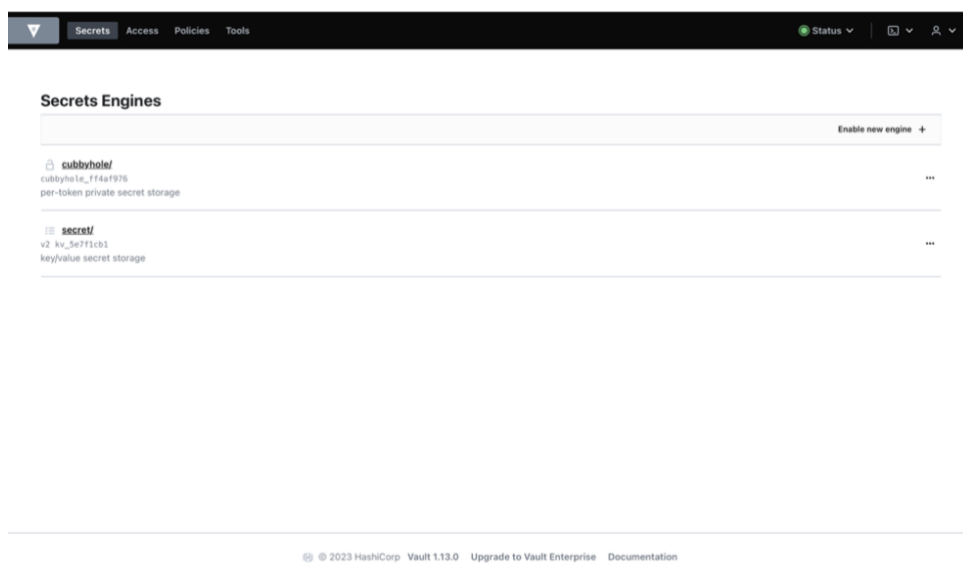


Ilustración 12. Dashboard de Vault



Más información en la documentación oficial:

<https://developer.hashicorp.com/vault/tutorials/kubernetes/kubernetes-minikube-raft>

b. Buenas prácticas Vault

Aquí tienes algunas buenas prácticas al utilizar Vault:

Buenas prácticas	Descripción
Seguir el principio de privilegio mínimo	Asignar los permisos mínimos necesarios a los usuarios y aplicaciones para acceder a los secretos y funcionalidades de Vault.
Implementar autenticación sólida	Utilizar métodos de autenticación seguros, como LDAP, Azure AD o JWT, para autenticar a los usuarios y aplicaciones en Vault.
Utilizar políticas de acceso granular	Definir políticas de acceso detalladas para controlar qué usuarios y aplicaciones pueden acceder a los secretos almacenados en Vault y qué operaciones pueden realizar.
Encriptar todos los datos en reposo y en tránsito	Configurar Vault para que todos los datos se almacenen encriptados y para que las comunicaciones se realicen a través de conexiones seguras (TLS).
Realizar copias de seguridad regulares	Realizar copias de seguridad periódicas de la base de datos y la configuración de Vault para poder restaurarla en caso de pérdida de datos o fallos del sistema.
Actualizar y parchear Vault regularmente	Mantener la instancia de Vault actualizada con las últimas versiones y parches de seguridad para evitar posibles vulnerabilidades conocidas.
Habilitar el registro y el monitoreo de auditoría	Configurar Vault para que registre y audite todas las actividades, incluidos los intentos de acceso, las operaciones realizadas y los cambios en la configuración.
Proteger la infraestructura que aloja Vault	Implementar medidas de seguridad adecuadas para proteger los servidores y la infraestructura que alojan Vault, como cortafuegos, autenticación multifactor y acceso seguro.
Realizar pruebas de penetración y auditorías de seguridad	Realizar pruebas periódicas de penetración y auditorías de seguridad para identificar posibles vulnerabilidades y evaluar la robustez de la configuración y la implementación de Vault.

Tabla 5. Buenas prácticas para la gestión de Vault



c. Comandos Vault

A continuación, se muestran los comandos más empleados en Vault.

Comando	Descripción
<code>vault server -dev</code>	Inicia un servidor de desarrollo de Vault en modo local para propósitos de pruebas y desarrollo.
<code>vault login</code>	Inicia sesión en Vault con las credenciales de autenticación correspondientes.
<code>vault secrets enable</code>	Habilita un motor de secretos específico en Vault para almacenar y administrar secretos.
<code>vault kv put</code>	Almacena un secreto en el motor de secretos de Vault.
<code>vault kv get</code>	Recupera un secreto específico del motor de secretos de Vault.
<code>vault kv delete</code>	Elimina un secreto del motor de secretos de Vault.
<code>vault policy write</code>	Crea o actualiza una política de acceso en Vault para controlar los permisos de los usuarios y aplicaciones.
<code>vault token create</code>	Genera un nuevo token de acceso en Vault con los permisos adecuados.
<code>vault status</code>	Muestra el estado actual de Vault.
<code>vault operator init</code>	Inicializa un clúster de Vault con claves de encriptación y claves de desbloqueo.

Tabla 6. Comandos útiles para gestionar Vault

5.1.2.2 Sealed Secrets

Sealed Secrets es una herramienta que permite cifrar y almacenar de forma segura secretos sensibles en un clúster de Kubernetes. Utiliza criptografía asimétrica para proteger los secretos y garantizar que solo los destinatarios autorizados puedan descifrarlos. Los secretos sellados se almacenan como objetos de Kubernetes en un repositorio de código fuente, lo que facilita su gestión y seguimiento mediante herramientas de control de versiones. Esta solución es especialmente útil para entornos en los que se requiere el almacenamiento seguro de secretos, como credenciales de bases de datos, claves de API y certificados SSL/TLS.



a. Instalación Sealed Secrets en minikube



20 min



Intermedio

A continuación, se muestran los pasos a seguir para instalar Sealed Secrets en minikube:

1. Instalación del Controlador

PASO 1. Agrega el repositorio de Helm de Sealed Secrets ejecutando el siguiente comando:

```
$ helm repo add sealed-secrets https://bitnami-labs.github.io/sealed-secrets
```

PASO 2. Al ejecutar el comando, asegúrate de que el nombre del controlador (sealed-secrets) coincida con el nombre del controlador de Sealed Secrets que has instalado en el clúster de Kubernetes. Esto permitirá que kubeseal se comunique correctamente con el controlador y realice las operaciones necesarias en los secretos sellados.

```
$ kubeseal --controller-name sealed-secrets
```

PASO 4. Como alternativa, puedes establecer "fullnameOverride" al instalar el chart para cambiar el nombre. También ten en cuenta que "kubeseal" asume que el controlador está instalado dentro del espacio de nombres "Kube-system" de forma predeterminada. Por lo tanto, si deseas utilizar la CLI de "kubeseal" sin tener que pasar el nombre del controlador y el espacio de nombres esperados, debes instalar el Helm Chart de la siguiente manera:

```
$ helm install sealed-secrets -n kube-system --set-string fullnameOverride=sealed-secrets-controller sealed-secrets/sealed-secrets
```

2. Instalación del cliente

Para instalar el cliente es necesario ejecutar el siguiente comando:

```
$ brew install kubeseal
```

Más información en el GitHub de colaboración o la documentación oficial:

<https://onthedock.github.io/post/210819-sealed-secrets/>

<https://github.com/bitnami-labs/sealed-secrets#usage>



b. Buenas prácticas

Esta herramienta se utiliza en entornos de Kubernetes para la gestión segura de secretos en forma encriptada.

Buena Práctica	Descripción
Limita el acceso a los Sealed Secrets	Restringe el acceso a los Sealed Secrets solo a los usuarios o servicios autorizados que necesiten utilizarlos. Esto ayuda a prevenir la exposición no autorizada de los secretos encriptados.
Protege la clave de encriptación	Asegúrate de que la clave de encriptación utilizada por Sealed Secrets esté debidamente protegida. Almacénala en un lugar seguro y restringe el acceso a la misma. Esto garantiza la integridad y confidencialidad de los secretos encriptados.
Realiza copias de seguridad de los Sealed Secrets	Implementa un proceso de copia de seguridad regular de los Sealed Secrets. Esto garantiza la disponibilidad de los secretos en caso de pérdida accidental o corrupción. Asegúrate de almacenar las copias de seguridad en un lugar seguro y protegido.
Limita los permisos de acceso a los Sealed Secrets	Otorga permisos de acceso a los Sealed Secrets únicamente a los usuarios o servicios necesarios. Utiliza el principio de menor privilegio para minimizar el riesgo de exposición indebida.
Implementa políticas de rotación de claves	Establece un proceso de rotación periódica de las claves de encriptación utilizadas por Sealed Secrets. Esto ayuda a mantener la seguridad de los secretos encriptados y limita la exposición en caso de una posible violación de seguridad.
Realiza pruebas de recuperación	Periódicamente, realiza pruebas de recuperación para asegurarte de que puedes restaurar y descryptar correctamente los Sealed Secrets en caso de pérdida o corrupción. Esto garantiza la disponibilidad de los secretos en situaciones de emergencia.
Documentación y comunicación de las políticas de uso	Documentar claramente las políticas y procedimientos relacionados con el uso de Sealed Secrets, y asegurarse de que sean comunicados y comprendidos por todos los usuarios involucrados. Esto promueve un uso consistente y seguro de la herramienta.

Tabla 7. Buenas prácticas para la gestión de Sealed Secrets

Estas buenas prácticas ayudarán a utilizar Sealed Secrets de manera segura y efectiva en el entorno de Kubernetes. Recuerda adaptarlas a las necesidades y requisitos específicos de seguridad de la organización.



c. Comandos Sealed Secrets

A continuación, se presenta una tabla con algunos de los comandos más utilizados en Sealed Secrets:

Comando	Descripción
kubeseal --fetch-cert	Descarga el certificado público del controlador Sealed Secrets y lo guarda en un archivo local.
kubeseal --cert cert.pem < secret.yaml	Sella un archivo YAML con los secretos proporcionados y muestra el resultado sellado en la salida estándar.
kubeseal --reencrypt	Vuelve a cifrar todos los secretos sellados utilizando el certificado actual.
kubeseal --unseal	Descifra un secreto sellado y muestra los secretos descifrados en la salida estándar.

Tabla 8. Comandos útiles para Sealed Secrets

Estos comandos son algunos de los más comunes al trabajar con Sealed Secrets. Recuerda reemplazar <cert.pem> por el nombre del archivo de certificado descargado y <secret.yaml> por el nombre del archivo YAML que deseas sellar o descifrar.




5.1.3 A nivel de gestión de políticas

5.1.3.1 Kyverno

Kyverno es una herramienta de gestión de políticas para clústeres de Kubernetes. Permite definir y aplicar políticas de manera declarativa para asegurar el cumplimiento de reglas y mejores prácticas en la configuración y el despliegue de recursos en el clúster. Kyverno se integra directamente con Kubernetes y puede evaluar y modificar los objetos de Kubernetes en tiempo real. Proporciona un enfoque basado en políticas para el control de acceso, la validación de estructura, la configuración de seguridad y otros aspectos relacionados con la gobernanza y el cumplimiento. Kyverno simplifica la gestión de políticas en Kubernetes y ayuda a garantizar que el clúster esté configurado de acuerdo con los estándares deseados. Existen 4 tipos de políticas en kyverno lo cual hace muy interesante esta herramienta frente a OPA. En la documentación oficial se muestra más información acerca de ello.

a. Instalación Kyverno

 15min  Fácil

A continuación, se muestran los pasos a seguir para instalar Kyverno en Minikube:

PASO 1. Abrir minikube

```
$ minikube start
```

PASO 2. Instalar Kyverno

- Opción 1. Con kubectl

```
$ kubectl create -f https://raw.githubusercontent.com/kyverno/kyverno/main/config/install.yaml
```

- Opción 2. Con helm

```
$ helm repo add kyverno https://kyverno.github.io/kyverno/  
$ helm repo update  
# Instalar controlador de kyverno  
$ helm install kyverno kyverno/kyverno --namespace kyverno --create-namespace  
# Instalar políticas de kyverno  
helm install kyverno-policies kyverno/kyverno-policies --namespace Kyverno
```

PASO 3. Para verificar el estado del controlador de Kyverno, ejecuta el siguiente comando:

```
$ kubectl get pods -n <namespace>
```



Si el controlador de Kyverno no se está ejecutando, puedes verificar su estado y los registros en busca de errores:

```
$ kubectl describe pod <kyverno-pod-name> -n <namespace>
$ kubectl logs -l app.kubernetes.io/name=kyverno -n <namespace>
```

Más información en el GitHub de colaboración o la documentación oficial:

<https://kyverno.io/docs/installation/methods/>
<https://dev.to/kcdchennai/how-to-install-kyverno-in-a-k8s-cluster-1g20>

b. Comandos Kyverno

A continuación, se muestra una tabla de los principales comandos de Kyverno:

Comando	Descripción
<code>kubectl kyverno apply</code>	Aplicar las políticas de Kyverno a un clúster de Kubernetes utilizando el cliente kubectl
<code>kubectl kyverno get policies</code>	Obtener una lista de todas las políticas de Kyverno presentes en el clúster de Kubernetes.
<code>kubectl kyverno describe policy <nombre></code>	Mostrar información detallada sobre una política específica de Kyverno.
<code>kubectl kyverno create policy -f <archivo></code>	Crear una política de Kyverno utilizando un archivo YAML o JSON como fuente.
<code>kubectl kyverno delete policy <nombre></code>	Eliminar una política de Kyverno específica del clúster de Kubernetes.
<code>kubectl kyverno generate <directorío></code>	Generar políticas de Kyverno en base a los recursos existentes en un directorio de Kubernetes.
<code>kubectl kyverno apply cluster-policy -f <archivo></code>	Aplicar una política de Kyverno a nivel de clúster utilizando un archivo YAML o JSON como fuente.
<code>kubectl kyverno validate <directorío></code>	Validar los recursos de Kubernetes en un directorio específico utilizando las políticas de Kyverno definidas.
<code>kubectl kyverno explain <recurso></code>	Proporcionar información detallada sobre cómo una política de Kyverno afectará a un recurso específico de Kubernetes.
<code>kubectl kyverno show policy-report</code>	Mostrar un informe detallado de las políticas de Kyverno y su estado de cumplimiento en el clúster de Kubernetes.
<code>kubectl kyverno version</code>	Mostrar la versión actual de Kyverno instalada en el clúster de Kubernetes.

Tabla 9. Comandos útiles para Kyverno

c. Buenas prácticas



A continuación, se muestra una tabla con algunas buenas prácticas al utilizar la herramienta Kyverno para la validación y mutación de políticas en Kubernetes:

Buenas prácticas	Descripción
Comprende los conceptos fundamentales de Kyverno	Familiarizarse con los conceptos clave de Kyverno, como reglas, condiciones, mutaciones y políticas. Esto te permitirá utilizar la herramienta de manera efectiva y comprender cómo se aplican las políticas en el clúster de Kubernetes.
Define políticas específicas y granulares	Crear políticas que sean específicas y enfocadas en un conjunto particular de recursos. Evita políticas genéricas que abarquen un amplio rango de recursos, ya que esto puede generar conflictos y dificultar el mantenimiento.
Valida las políticas antes de aplicarlas	Antes de aplicar una política en el clúster de Kubernetes, valida su sintaxis y verifica su comportamiento esperado. Esto ayudará a identificar posibles errores o impactos no deseados en los recursos.
Utiliza anotaciones para aplicar políticas selectivamente	Utilizar anotaciones en los recursos de Kubernetes para aplicar políticas de Kyverno de manera selectiva. Esto permite controlar qué recursos son afectados por cada política, brindando flexibilidad y granularidad en la aplicación de políticas.
Documenta y comunica las políticas	Documenta claramente las políticas que definidas en Kyverno, incluyendo su propósito, reglas y cualquier consideración especial. Comunica esta documentación a los miembros relevantes del equipo para asegurar una comprensión adecuada y un uso consistente de las políticas.
Realiza pruebas exhaustivas de las políticas	Antes de implementar políticas de Kyverno en un entorno de producción, realiza pruebas exhaustivas en un entorno de desarrollo o de pruebas. Asegúrate de que las políticas se apliquen correctamente y no tengan efectos no deseados en los recursos.
Monitorea y revisa las violaciones de políticas	Establecer un monitoreo continuo de las violaciones de políticas detectadas por Kyverno. Revisar regularmente los registros y las notificaciones generadas por Kyverno para identificar y abordar las violaciones de manera oportuna.
Mantén las políticas actualizadas	A medida que la infraestructura de Kubernetes evolucione, revisar y actualizar las políticas de Kyverno en consecuencia. Esto garantiza que las políticas sigan siendo relevantes y efectivas para mantener la seguridad y el cumplimiento del clúster.

Tabla 10. Buenas prácticas de la gestión de Kyverno



d. Ejemplo en Kyverno

En la documentación oficial se muestran diversos ejemplos bien detallados:

<https://kyverno.io/docs/kyverno-cli/>

5.1.3.2 OPA - Gatekeeper

OPA (Open Policy Agent) Gatekeeper es una herramienta de cumplimiento y gobernanza para clústeres de Kubernetes. Permite definir y hacer cumplir políticas de seguridad y reglas personalizadas para los recursos desplegados en el clúster. Gatekeeper utiliza OPA, un motor de políticas flexible y extensible, para evaluar y aplicar políticas declarativas en tiempo real. Permite establecer restricciones y controles sobre los recursos de Kubernetes, como pod, deployment, service, etc., para garantizar la conformidad con los estándares de seguridad y las mejores prácticas establecidas. OPA Gatekeeper ayuda a mantener un entorno de Kubernetes seguro y garantiza que los recursos se creen y gestionen de acuerdo con las políticas definidas.

a. Instalación Kyverno

 15 min  Fácil

A continuación, se muestran los pasos a seguir para instalar Kyverno.

PASO 1. Abrir minikube

```
$ minikube start
```

PASO 2. Instalar OPA GateKeeper

- Opción 1. Instalación con kubectl

```
$ kubectl apply -f https://raw.githubusercontent.com/open-policy-agent/gatekeeper/master/deploy/gatekeeper.yaml
```

- Opción 2. Instalación con helm

```
$ helm repo add gatekeeper https://open-policy-agent.github.io/gatekeeper/charts  
$ helm install gatekeeper/gatekeeper --name-template=gatekeeper --namespace gatekeeper-system --create-namespace
```

PASO 3. Después del despliegue, se creará un nuevo espacio de nombres (namespace) llamado gatekeeper-system y se crearán los siguientes recursos:

```
$ kubectl get deployments  
$ kubectl get services  
$ kubectl get crd
```

Más información en el GitHub de colaboración o la documentación oficial:



<https://open-policy-agent.github.io/gatekeeper/website/docs/install/>

<https://docs.mirantis.com/mke/3.6/ops/deploy-apps-k8s/policy-enforcement/deploy-gatekeeper/install-gatekeeper.html>

b. Comandos OPA

A continuación, se muestra algunos comandos

Comando	Descripción
<code>kubectl apply -f <restricciones.yaml></code>	Aplica las restricciones definidas en un archivo YAML al clúster de Kubernetes utilizando el cliente kubectl.
<code>kubectl get audit -l gatekeeper.sh/Audit</code>	Obtiene los registros de auditoría generados por OPA Gatekeeper para evaluar el cumplimiento de las restricciones.
<code>kubectl gatekeeper stats</code>	Muestras estadísticas y métricas relacionadas con las restricciones y violaciones en OPA Gatekeeper.
<code>kubectl gatekeeper health</code>	Verifica el estado de salud de OPA Gatekeeper y asegura que esté funcionando correctamente.
<code>kubectl gatekeeper versión</code>	Muestra la versión actual de OPA Gatekeeper instalada en el clúster de Kubernetes.

Tabla 11. Comandos útiles OPA Gatekeeper

c. Buenas prácticas OPA

Se muestra una tabla con algunas buenas prácticas al utilizar la herramienta OPA Gatekeeper para implementar políticas de seguridad en Kubernetes:

Buena Práctica	Descripción
Comprende los conceptos de OPA Gatekeeper	Familiarizarse con los conceptos clave de OPA Gatekeeper, como restricciones, plantillas, evaluaciones y auditorías. Esto te permitirá utilizar la herramienta de manera efectiva y entender cómo se aplican las políticas en el clúster de Kubernetes.
Diseña políticas de seguridad adecuadas	Definir políticas de seguridad que sean específicas y se ajusten a los requisitos del entorno. Considerar los estándares de seguridad relevantes y las mejores prácticas de la industria para garantizar que las políticas sean sólidas y eficaces.
Valida y prueba las políticas antes de implementarlas	Antes de implementar una política en el clúster de Kubernetes, validar la sintaxis y realizar pruebas exhaustivas en un entorno de



	desarrollo o de pruebas. Asegurarse de que las políticas se apliquen correctamente y no tengan impactos inesperados en los recursos.
Utiliza restricciones predefinidas o restricciones personalizadas	Aprovecha las restricciones predefinidas de OPA Gatekeeper para implementar políticas comunes de seguridad. Si es necesario, crear restricciones personalizadas que se ajusten a los requisitos específicos del entorno.
Implementa auditorías y evaluaciones periódicas	Configurar auditorías y evaluaciones periódicas para asegurarse de que las políticas se apliquen de manera continua y efectiva en el clúster de Kubernetes. Monitorear y revisar los registros y las notificaciones generadas por OPA Gatekeeper para identificar y abordar las violaciones de políticas.
Documenta las políticas y su propósito	Documentar claramente las políticas implementadas en OPA Gatekeeper, incluyendo su propósito, las restricciones aplicadas y cualquier consideración especial. Comunica esta documentación a los miembros relevantes del equipo para garantizar una comprensión adecuada y un uso consistente de las políticas.
Mantén las políticas actualizadas	A medida que la infraestructura de Kubernetes evolucione, revisa y actualiza las políticas en OPA Gatekeeper en consecuencia. Esto garantiza que las políticas sigan siendo relevantes y efectivas para mantener la seguridad y el cumplimiento en el clúster.
Implementa políticas en un entorno de desarrollo o pruebas primero	Antes de implementar políticas de OPA Gatekeeper en un entorno de producción, desplegarlas y probarlas en un entorno de desarrollo o pruebas. Esto permite validar el funcionamiento y abordar cualquier problema antes de afectar el entorno de producción.

Tabla 12. Buenas prácticas de la gestión de OPA GateKeeper



5.2 Instalación herramientas a nivel de tiempo de ejecución en contenedores

5.2.1 A nivel de Escaneo de imágenes

5.2.1.1 Trivy

Trivy es una herramienta de escaneo de seguridad diseñada para analizar imágenes de contenedores en busca de vulnerabilidades. Utiliza una amplia base de datos de vulnerabilidades conocidas y realiza un escaneo exhaustivo de las capas de las imágenes para identificar cualquier riesgo potencial. Trivy es fácil de usar y se puede integrar en los procesos de construcción y despliegue de imágenes de contenedores. Proporciona informes detallados sobre las vulnerabilidades encontradas, incluyendo su gravedad y recomendaciones para su solución. Trivy es una herramienta útil para garantizar la seguridad de las imágenes de contenedores utilizadas en entornos de desarrollo y producción, y ayuda a prevenir posibles ataques y brechas de seguridad.

a. Instalación Trivy en Minikube



15min



Fácil

A continuación, se muestran los pasos a seguir para instalar Trivy.

Requisitos

- Entorno Kubernetes
- Kubectl or Helm or Homebrew

Instalación Trivy

Se ofrecen 3 modos de instalación de Trivy:

- Opción 1

Para instalar Trivy en Minikube, puedes seguir estos pasos:

Paso 1. Inicia el clúster Minikube ejecutando el siguiente comando en la terminal:

```
$ minikube start
```

Paso 2. Descarga el archivo YAML de despliegue de Trivy ejecutando el siguiente comando:

```
$ curl -LO https://github.com/aquasecurity/trivy/releases/latest/download/trivy_deploy_minikube.yaml
```

Paso 3. Aplica el archivo YAML de despliegue para crear los recursos de Trivy en el clúster Minikube:

```
$ kubectl apply -f trivy_deploy_minikube.yaml
```

Paso 4. Verifica que el pod de Trivy esté en funcionamiento ejecutando el siguiente comando:

```
$ kubectl get pods -n trivy
```



Deberías ver el pod de Trivy en estado "Running" y listo para realizar análisis de seguridad. Ahora estás listo para usar Trivy en el clúster Minikube.

- Opción 2

Para instalar Trivy con Homebrew en el dispositivo:

Paso 1. You can use homebrew on macOS and Linux.

```
$ brew install aquasecurity/trivy/trivy
```

- Opción 3

Para instalar Trivy con Helm en minikube:

Paso 1. Añadir el repositorio de Trivy:

```
$ helm repo add aquasecurity https://aquasecurity.github.io/helm-charts/
```

Paso 2. Actualizamos por si acaso el repositorio

```
$ helm repo update
```

Paso 3. Instalamos el repositorio

```
$ helm install my-trivy aquasecurity/trivy
```

- Opción 4

Paso 1. Instalación con kubectl sin necesidad de descargar todos los archivos. Esto instalará el operador en el espacio de nombres trivy-system y lo configurará para escanear todos los espacios de nombres, excepto kube-system y trivy-system:

```
$ kubectl apply -f https://raw.githubusercontent.com/aquasecurity/trivy-operator/v0.0.3/deploy/static/trivy-operator.yaml
```

Paso 2. Para confirmar que el operador está en funcionamiento, verifica que el Despliegue (Deployment) trivy-operator en el espacio de nombres trivy-system esté disponible y que todos sus contenedores estén listos:

```
$ kubectl get deployment -n trivy-system
```

Si por alguna razón aún no está listo, verifica los registros (logs) del Despliegue trivy-operator en el espacio de nombres trivy-system en busca de errores:

```
$ kubectl logs deployment/trivy-operator -n trivy-system
```

Esto permitirá verificar el estado del operador y solucionar cualquier problema si es necesario.

Más información en la documentación oficial:

<https://aquasecurity.github.io/trivy/v0.18.3/installation/>

<https://aquasecurity.github.io/trivy/v0.28.1/docs/kubernetes/operator/installation/kubectl/>

<https://aquasecurity.github.io/trivy/v0.28.1/docs/kubernetes/cli/scanning/>

<https://aquasecurity.github.io/trivy/v0.28.1/docs/kubernetes/operator/getting-started/>



b. Comandos Trivy

A continuación, se muestra una tabla con algunos comandos interesantes de Trivy:

Comando	Descripción
trivy image <nombre_de_imagen>	Escanea una imagen de contenedor en busca de vulnerabilidades.
trivy filesystem <ruta_al_directorio>	Escanea un directorio del sistema de archivos en busca de vulnerabilidades.
trivy repository <nombre_del_repositorio>	Escanea un repositorio de contenedores en busca de vulnerabilidades.
trivy client --stop-server	Detiene el servidor Trivy si está en ejecución.
trivy image --clear-cache	Borra la caché local utilizada por Trivy para las imágenes escaneadas.
trivy report --format <formato>	Genera un informe de escaneo en el formato especificado.
trivy db update	Actualiza la base de datos de vulnerabilidades utilizada por Trivy.

Tabla 13. Comandos útiles en Trivy

b. Buenas prácticas Trivy

A continuación, se muestra una tabla con las buenas prácticas de Trivy

Buena Práctica	Descripción
Actualiza Trivy regularmente	Mantener la versión de Trivy actualizada para aprovechar las últimas mejoras, correcciones de errores y bases de datos de vulnerabilidades actualizadas. Esto ayudará a mantener los análisis y escaneos de contenedores eficientes y precisos.
Escanea las imágenes de contenedor antes de implementarlas	Realizar análisis y escaneos de vulnerabilidades en las imágenes de contenedor antes de implementarlas en un entorno de producción. Esto permite identificar y abordar posibles vulnerabilidades antes de que los contenedores se ejecuten en producción.



Configura el nivel de severidad adecuado	Ajustar el nivel de severidad del escaneo de Trivy en función de las necesidades y prioridades. Puedes configurarlo para que se centren en las vulnerabilidades críticas o también incluir vulnerabilidades de menor severidad según las políticas de seguridad.
Realiza escaneos periódicos y automáticos	Establecer escaneos automáticos y periódicos con Trivy para garantizar que las imágenes de contenedor se mantengan actualizadas y asegurar a lo largo del tiempo. Esto ayuda a identificar nuevas vulnerabilidades que puedan surgir en las imágenes actualizadas.
Integra Trivy en los pipelines de CI/CD	Incorpora Trivy en los pipelines de integración y entrega continuas (CI/CD) para realizar escaneos automáticos durante el proceso de construcción de imágenes y antes de implementarlas en producción. Esto garantiza que las imágenes sean seguras desde el inicio y evita la introducción de vulnerabilidades en el proceso de desarrollo.
Utiliza las opciones de salida adecuadas	Aprovecha las opciones de salida de Trivy para obtener informes y resultados claros y comprensibles. Así poder generar informes en formatos como JSON, CSV o incluso integrarlos con otras herramientas o sistemas de gestión de vulnerabilidades.
Implementa correcciones y actualizaciones	Una vez que Trivy identifique vulnerabilidades en las imágenes del contenedor, tomar medidas correctivas aplicando parches o actualizando las dependencias afectadas. Asegurarse de seguir las mejores prácticas de gestión de vulnerabilidades y de abordar las vulnerabilidades de mayor gravedad primero.
Documenta y comunica los resultados	Documentar los resultados de los escaneos realizados por Trivy, incluyendo las vulnerabilidades encontradas y las acciones tomadas para solucionarlas. Comunica estos resultados a los miembros relevantes del equipo y asegurarse de que se tomen las medidas adecuadas para abordar las vulnerabilidades identificadas.

Tabla 14. Buenas prácticas de la gestión de Trivy

c. Ejemplo Trivy

Puedes ejecutar análisis de seguridad en las imágenes de contenedor usando el siguiente comando. Creamos la implementación (Deployment) de nginx que sabemos que es vulnerable:

```
$ kubectl create deployment nginx --image nginx:1.16
```

Cuando se crea la implementación (Deployment) de nginx, el operador detecta de inmediato su revisión actual (también conocida como ReplicaSet activo) y escanea la imagen nginx:1.16 en



busca de vulnerabilidades. También audita la especificación del ReplicaSet en busca de problemas comunes, como ejecutar el contenedor nginx como root.

Si todo va bien, el operador guarda los informes de escaneo como recursos VulnerabilityReport y ConfigAuditReport en el namespace predeterminado. Los informes reciben nombres basados en el ReplicaSet escaneado. Para los escaneos de vulnerabilidades de la imagen, el operador crea un VulnerabilityReport para cada contenedor diferente. En este ejemplo, solo hay una imagen de contenedor llamada nginx.

```
$ kubectl get vulnerabilityreports -o wide
```

O por ejemplo:

```
$ trivy image nginx:latest
```

Trivy escaneará la imagen y proporcionará los resultados de las vulnerabilidades encontradas.

5.2.1.2 Clair

Clair es una herramienta de escaneo de seguridad específicamente diseñada para imágenes de contenedores. Su objetivo principal es identificar y notificar sobre posibles vulnerabilidades presentes en las imágenes de contenedores utilizadas en un entorno de Kubernetes. Clair utiliza una base de datos de vulnerabilidades conocidas y realiza análisis exhaustivos de las capas de las imágenes para detectar cualquier riesgo de seguridad. Proporciona informes detallados sobre las vulnerabilidades encontradas, incluyendo información sobre su gravedad y posibles soluciones. Con Clair, los equipos de seguridad pueden tomar medidas proactivas para mitigar los riesgos de seguridad en las imágenes de contenedores y garantizar que se implementen las mejores prácticas de seguridad en el clúster de Kubernetes.

a. Instalación Clair en Minikube

 20min  Fácil

A continuación, se muestran los pasos a seguir para instalar Clair.

Requisitos

- Entorno Kubernetes: Minikube

Instalación Clair

Paso 1. Inicia el clúster de Minikube ejecutando el siguiente comando en la terminal:

```
$ minikube start
```

Paso 2. Descarga el archivo YAML de la configuración de Clair. Puedes obtenerlo desde el repositorio oficial de Clair en GitHub o utilizando el siguiente comando:

```
$ curl -LO https://raw.githubusercontent.com/coreos/clair/master/contrib/k8s/clair.yaml
```



Paso 3. Abre el archivo `clair.yaml` en un editor de texto y realiza cualquier configuración adicional que desees, como cambiar el nombre del servicio o el número de réplicas.

Aplica la configuración de Clair en los clústeres de Minikube ejecutando el siguiente comando:

```
$ kubectl apply -f clair.yaml
```

Paso 4. Verifica que los pods de Clair se estén ejecutando correctamente utilizando el siguiente comando:

```
$ kubectl get pods
```

Paso 5. Asegúrate de que los pods de Clair estén en estado "Running" antes de continuar. Una vez que Clair esté instalado y en funcionamiento en el clúster de Minikube, se puede comenzar a utilizarlo para escanear imágenes de contenedor y obtener informes de vulnerabilidades.

Más información en el GitHub de colaboración o la documentación oficial:

<https://github.com/quay/clair>

b. Buenas prácticas Clair

A continuación, se muestra algunas buenas prácticas al utilizar la herramienta Clair para el análisis de vulnerabilidades en imágenes de contenedor.

Buena Práctica	Descripción
Actualiza Clair regularmente	Mantén la versión de Clair actualizada para aprovechar las últimas mejoras, correcciones de errores y bases de datos de vulnerabilidades actualizadas. Esto ayudará a obtener resultados precisos y confiables al analizar las imágenes de contenedor.
Escanea las imágenes de contenedor antes de implementarlas	Realizar análisis de vulnerabilidades en las imágenes de contenedor utilizando Clair antes de implementarlas en producción. Esto permite identificar y abordar posibles vulnerabilidades antes de que los contenedores se ejecuten en un entorno en vivo.
Configura el nivel de severidad adecuado	Ajustar el nivel de severidad del escaneo de Clair en función de las necesidades y prioridades. Puedes enfocarte en vulnerabilidades críticas o también incluir vulnerabilidades de menor severidad, según las políticas de seguridad establecidas.
Realiza escaneos periódicos y automáticos	Establecer escaneos automáticos y periódicos con Clair para garantizar que las imágenes de contenedor se mantengan actualizadas y seguras a lo largo del tiempo. Esto ayuda a identificar nuevas vulnerabilidades que puedan surgir en las imágenes actualizadas.



Integra Clair con Harbor	Aprovechar la integración entre Clair y Harbor para obtener una solución completa de gestión de imágenes y seguridad. Utilizar Harbor para organizar y almacenar las imágenes de contenedor, y Clair para analizar y detectar vulnerabilidades en esas imágenes.
Implementa correcciones y actualizaciones	Una vez que Clair identifique vulnerabilidades en las imágenes de contenedor, tomar medidas correctivas aplicando parches o actualizando las dependencias afectadas. Asegurarse de seguir las mejores prácticas de gestión de vulnerabilidades y de abordar las vulnerabilidades críticas primero.
Documenta y comunica los resultados	Documentar los resultados de los escaneos realizados por Clair, incluyendo las vulnerabilidades encontradas y las acciones tomadas para solucionarlas. Comunica estos resultados a los miembros relevantes del equipo y asegúrate de que se tomen las medidas adecuadas para abordar las vulnerabilidades identificadas.
Realiza auditorías y revisiones regulares	Llevar a cabo auditorías y revisiones periódicas de las imágenes de contenedor utilizando Clair y Harbor para garantizar la conformidad con las políticas de seguridad establecidas. Revisa y actualiza las imágenes según sea necesario para mantener un entorno seguro y protegido.

Tabla 15. Buenas prácticas de la gestión de Clair

Estas buenas prácticas ayudarán a utilizar Clair de manera efectiva y a integrarlo con Harbor para organizar y proteger las imágenes de contenedor.



5.2.2 A nivel de tiempo de ejecución

5.2.2.1 Falco

Falco es una herramienta de detección de seguridad y prevención de intrusiones diseñada específicamente para entornos de contenedores y orquestadores como Kubernetes. Utiliza reglas y políticas personalizables para monitorear y detectar actividades maliciosas y comportamientos anómalos en tiempo real dentro del clúster. Falco se basa en eventos del kernel y en la trazabilidad de las llamadas al sistema para identificar actividades sospechosas, como intentos de acceso no autorizados, comportamientos anómalos de procesos y violaciones de políticas de seguridad. Cuando se detecta una actividad sospechosa, Falco puede generar alertas en tiempo real y tomar medidas para prevenir posibles ataques o violaciones de seguridad.

A medida que Kubernetes crece en adopción, es crucial saber cómo asegurarlo. En una plataforma de infraestructura dinámica como Kubernetes, detectar y abordar amenazas es un desafío. Falco es uno de los principales motores de detección de amenazas de Kubernetes de código abierto.

La forma recomendada de ejecutar Falco es instalarlo directamente en el sistema host. Esto aísla a Falco de Kubernetes en caso de compromiso, y las alertas de Falco se pueden consumir a través de los agentes que se ejecutan en Kubernetes. Falco también se puede ejecutar directamente en Kubernetes si el aislamiento no es una preocupación. Sin embargo, para esta guía se instalará directamente en Kubernetes.

a. Instalación Falco en Minikube



30min



Fácil

Este enfoque asume que el aislamiento no es una preocupación y utiliza Helm para configurar Falco. Se puede implementar Falco en pocos segundos utilizando el repositorio de charts respaldado por la comunidad si ya estás utilizando Helm para administrar Kubernetes. A continuación, se muestran los pasos para instalar Falco en Minikube.

Requisitos

- Entorno Kubernetes: Minikube

Implementación

El chart de Helm utilizado en este enfoque utiliza un DaemonSet para agregar Falco a todos los nodos del clúster. Se implementa un pod de Falco en cada nodo para monitorear cualquier comportamiento anormal.

Hay cuatro pasos principales para instalar y ejecutar correctamente Falco en el sistema host:

- Instalar los encabezados del kernel.
- Agregar el repositorio del chart de Falco.



- c. Instalar el chart.
- d. Comprobar Falco.

Instalación Falco

Paso 1. Ejecutar Minikube:

```
$ minikube start
```

Paso 2. Comprobar rápidamente que se este ejecutando correctamente el entorno:

```
$ minikube kubectl -- get nodes
```

Paso 3. Instalar los encabezados del kernel

Verifica que los encabezados del kernel no estén instalados.

```
$ apt search linux-headers-$(uname -r)
```

Ejecuta el siguiente comando para instalar los encabezados del kernel:

```
$ apt-get -y install linux-headers-$(uname -r)
```

Verifica que los encabezados del kernel estén instalados ahora.

```
$ apt search linux-headers-$(uname -r)
```

Paso 4. Agregar el repositorio del chart de Falco

Antes de instalar el chart, agrega el repositorio de charts de falcosecurity:

```
$ helm repo add falcosecurity https://falcosecurity.github.io/charts
```

```
$ helm repo update
```

Paso 5. Instalar el chart:

Ejecuta el siguiente comando para crear un espacio de nombres (namespace) para Falco e instalar el chart de Falco. Se recomienda usar una versión específica del chart de Helm y mantenerla en futuras actualizaciones del release de Helm:

```
$ kubectl create ns falco
```

```
$ helm install falco
```

```
-n falco
```

```
--version 2.4.2
```

```
--set tty=true
```

```
--set collectors.containerd.enabled=true
```

```
--set collectors.containerd.socket=/run/k3s/containerd/containerd.sock
```

```
falcosecurity/falco
```

O con el siguiente comando:

```
$ helm install falco falcosecurity/falco
```



Este entorno utiliza contenedores en lugar de docker, por lo que el colector está disponible en un socket diferente (configurado con las opciones de implementación de Helm `collectors.containerd` mencionadas anteriormente).

Paso 6. Espere unos segundos hasta que los contenedores estén iniciados.

```
$ kubectl wait pods --for=condition=Ready -l app.kubernetes.io/name=falco -n falco --timeout=150s
```

O con el siguiente commando:

```
$ kubectl get pods
```

Paso 7. Paso Una vez que el pod esté listo. Identifica el pod de Falco y revisa los logs del contenedor. Ejecuta el siguiente comando para ver los registros (logs):

```
$ kubectl logs -l app.kubernetes.io/name=falco -n falco
```

O con el siguiente comando:

```
$ kubectl logs falco-XXXXX
```

Los registros confirman que Falco y sus reglas se han cargado correctamente.

Falco ya está desplegado.

Más información en el GitHub de colaboración o la documentación oficial:

<https://github.com/falcosecurity/falco>

<https://falco.org/docs/getting-started/installation/>

<https://sysdig.com/blog/intro-runtime-security-falco/>

Notas

Nota sobre la actualización de un chart de Helm: Los charts de Helm tienen versiones. Con cada cambio nuevo o nueva función agregada a Falco, se lanza un nuevo chart.

Para evitar cambios que puedan causar problemas, se recomienda usar una versión específica del chart de Helm tanto al usar los comandos `helm install` como `helm upgrade`. Cada vez que necesites realizar cambios en las instancias de Falco desplegadas con Helm, asegúrate de utilizar la opción `--reuse-values` para reutilizar cualquier opción utilizada en la instalación inicial.

La opción de versión utilizada en la instalación inicial no se reutiliza al aplicar la opción `--reuse-values`. Por lo tanto, al usar `helm upgrade --reuse-values`, es una buena práctica especificar también la versión. Luego, puedes agregar los cambios que desees, como reglas personalizadas



en un archivo values.yaml. No ejecutes el siguiente comando. Practicarás esto en los próximos laboratorios:

```
$ helm upgrade falco falcosecurity/falco
--reuse-values
--version 2.4.2
-f values.yaml
```

b. Buenas prácticas Falco

Se presenta una tabla algunas buenas prácticas al utilizar la herramienta Falco para la detección y prevención de amenazas en tiempo real en entornos de contenedores y Kubernetes. También se menciona la relevancia de un curso específico sobre Falco:

Buena Práctica	Descripción
Configurar reglas y políticas relevantes	Definir reglas y políticas de detección específicas para el entorno y aplicaciones. Asegurarse de abordar los escenarios de amenazas relevantes y adaptar las reglas según las necesidades de seguridad.
Mantener actualizado Falco	Actualizar regularmente la versión de Falco para aprovechar las últimas correcciones de seguridad, mejoras de rendimiento y nuevas funcionalidades. Esto ayudará a mantener el sistema protegido y en línea con los avances más recientes.
Configurar acciones y notificaciones adecuadas	Configurar las acciones apropiadas para cada regla de detección en Falco. Esto puede incluir enviar notificaciones, generar alertas o ejecutar acciones automatizadas, según el nivel de gravedad y las políticas de respuesta establecidas.
Realizar pruebas exhaustivas	Antes de implementar Falco en un entorno de producción, realizar pruebas rigurosas en un entorno de desarrollo o de pruebas. Asegurarse de que las reglas y las acciones se comporten según lo esperado y no tengan impactos no deseados en las operaciones.
Monitorizar y revisar los registros de Falco	Mantener un monitoreo continuo de los registros generados por Falco para detectar y analizar cualquier actividad sospechosa o violación de seguridad. Revisar y analizar regularmente los registros para identificar posibles amenazas y tomar medidas correctivas.
Integrar Falco con otras herramientas de seguridad	Considerar la integración de Falco con otras herramientas de seguridad, como sistemas SIEM (Security Information and Event Management) o soluciones de orquestación de seguridad. Esto permitirá tener una visión



	más completa de las amenazas y facilitará la respuesta y mitigación eficientes.
Documentar las reglas y políticas de Falco	Documentar claramente las reglas y políticas implementadas en Falco, incluyendo su propósito, condiciones de activación y acciones asociadas. Comunicar esta documentación a los miembros relevantes del equipo para garantizar una comprensión adecuada y un uso consistente de Falco.
Actualizar conocimientos con cursos específicos	Mantener actualizado sobre las mejores prácticas y el uso eficiente de Falco. El curso "Falco 101" de Sysdig es una opción interesante para aprender más sobre la herramienta y su implementación. Aprovecha estos recursos de capacitación para mejorar las habilidades y conocimientos en el uso de Falco.

Tabla 16. Buenas prácticas de la gestión de Falco

Recuerda que estas buenas prácticas son generales y debes adaptarlas a las necesidades y requisitos específicos del entorno y las aplicaciones. El curso "Falco 101" mencionado puede ser una valiosa fuente de información para profundizar en el conocimiento de Falco.

Interesante informarse en este curso: <https://learn.sysdig.com/falco-101>



6. Casos de uso- Tabla resumen

A continuación, se presenta una tabla que resume los casos de uso para cada una de las herramientas mencionadas:

Herramienta	Casos de Uso
Kube-bench	Verificar el cumplimiento de las mejores prácticas de seguridad en los entornos de Kubernetes.
Kube-hunter	Identificar y abordar posibles vulnerabilidades en los clústeres de Kubernetes.
Vault	Gestionar y proteger secretos y datos confidenciales en entornos de Kubernetes.
Sealed Secrets	Encriptar y proteger secretos sensibles almacenados en archivos YAML en clústeres de Kubernetes.
Kyverno	Aplicar políticas y reglas de validación en tiempo real para garantizar la seguridad y cumplimiento en Kubernetes.
OPA GateKeeper	Implementar políticas de seguridad y cumplimiento en los recursos de Kubernetes.
Trivy	Escanear y detectar vulnerabilidades en las imágenes de contenedor utilizadas en Kubernetes.
Clair	Escanear y analizar imágenes de contenedor en busca de vulnerabilidades y riesgos de seguridad.
Falco	Detectar y prevenir comportamientos anómalos y actividades maliciosas en tiempo real en Kubernetes.

Tabla 17. Casos de uso de las herramientas

Estas herramientas brindan una amplia gama de funcionalidades para abordar diferentes aspectos de la seguridad y cumplimiento en entornos de Kubernetes.