Lo que pasa es que asegurar Kubernetes

Guía completa de seguridad en Kubernetes en lenguaje sencillo - Muestra gratuita (Capítulo 1)



Luis Felipe Ariza Vesga

Lo que pasa es que asegurar Kubernetes

Lo que pasa es que asegurar Kubernetes

Guía completa de seguridad en Kubernetes en lenguaje sencillo - Muestra gratuita (Capítulo 1)

Luis Felipe Ariza Vesga

Lo que pasa es que asegurar Kubernetes
Guía completa de seguridad en Kubernetes en lenguaje sencillo - Muestra gratuita (*Capítulo 1*)

Copyright © Luis Felipe Ariza Vesga.
Todos los derechos reservados.

La información de este libro se distribuye "tal cual", sin garantía. Aunque se han tomado todas las precauciones en su preparación, ni el autor ni la editorial asumen responsabilidad alguna por pérdidas o daños derivados directa o indirectamente de su contenido.

ISBN: 9798163540372

Para el warrah, el zorro de las Malvinas (*Dusicyon australis*)*, el único mamífero terrestre nativo de esas islas. No temía a las personas porque nunca había tenido un depredador, y Charles Darwin, que lo vio en 1833, contó que los gauchos lo mataban al atardecer ofreciéndole un pedazo de carne con una mano y sosteniendo un cuchillo con la otra. Lo cazaron por su piel, los colonos que lo creían una amenaza para sus ovejas lo envenenaron, y en 1876 ya no quedaba ninguno; la UICN lo registra como extinto. Darwin lo había previsto: en pocos años, escribió, este zorro estaría en la misma lista que el dodo. Y para todo el que cuida un clúster: la confianza se gana con pruebas, nunca se regala en la puerta.*

Índice general

Convenciones de este libro **1**
Tipografía y referencias 1
Las cuatro cajas de aviso 2
Créditos y licencias de terceros 3

Prefacio **5**
De qué se trata este libro 7
De qué no se trata este libro 7
Cómo está organizado 8
Sobre los comandos y ejemplos de código 9
Cómo contactarnos 10
Agradecimientos 11

I Configuración del clúster (15%) 13

1 El reglamento de quién entra a cada torre **15**
1.1 El conjunto sin reglamento de visitas 15
1.2 Dos reglamentos, no uno 17
1.3 La condición que decide si el reglamento se cumple 22
1.4 La revisión antes del turno 24
1.5 Practique en el laboratorio 25
1.6 Resumen 26
Bibliografía 27

Convenciones de este libro

El pequeño sistema visual de este libro, demostrado en vivo, no solo descrito.

Antes de arrancar, media página sobre cómo está escrito este libro. Se lee en dos minutos y le ahorra confusiones después.

Este libro le habla de **usted**. No es formalidad ni distancia: es la manera respetuosa de dirigirse a alguien con quien se trabaja hombro a hombro.

Cada capítulo abre con una escena cotidiana. Esa escena no es adorno: **es** el concepto técnico contado de otra manera, y vuelve al final del capítulo, en el resumen, ya explicada con las palabras correctas. Si una explicación técnica no le queda clara, regrese a la escena del comienzo: casi siempre la respuesta está ahí en la analogía del día a día. Y las escenas no andan sueltas. Forman una sola historia, la del conjunto residencial Los Nogales, que abre sus puertas en el capítulo 1 sin reglamento, sin cámaras y con un celador que deja pasar a todo el mundo, y que capítulo a capítulo aprende a cuidarse.

La analogía está para traducir el lenguaje técnico a situaciones que usted ya conoce, no para hacerlo reír. Nunca reemplaza el rigor: después de ella viene la definición oficial, con su enlace a la fuente, y cada comando impreso en estas páginas se ejecutó antes en un clúster real.

Tipografía y referencias

Tipografía monoespaciada se usa para todo lo que usted teclearía o leería en una pantalla: nombres de objetos como `NetworkPolicy` y `Secret`, campos de YAML como `seccompProfile`, comandos como `trivy image` y `cosign verify`, y rutas como `/var/lib/kubelet/seccomp/profiles`.

Negrita marca la primera aparición de un término importante, y también las palabras que suelen aparecer tal cual en el enunciado de las tareas del examen.

Los **paréntesis angulares** marcan lo que usted debe reemplazar: en `trivy image <nombre-de-la-imagen>`, usted escribe el nombre real de la imagen y no escribe los paréntesis.

Los **ejemplos** siguen siempre el mismo orden: una línea que dice qué se va a ver, el archivo YAML cuando hace falta, el bloque con el comando tal como se escribió y el bloque con la salida que devolvió el clúster en esa ejecución. Cuando una salida depende del tiempo o de la versión, el libro lo dice.

Las **citas de la documentación oficial** aparecen integradas en el texto y llevan un enlace a la fuente la primera vez que se usan en el capítulo, para que usted pueda verificar cualquier afirmación con un clic. En esta edición, lo que la documentación dice en inglés aparece traducido al español entre comillas, sin repetir el original.

Cada capítulo cierra con una **bibliografía** de al menos tres fuentes oficiales, y las **figuras** se citan en el texto que las explica.

Las cuatro cajas de aviso

A lo largo del libro va a encontrar cuatro tipos de caja. Cada una significa algo distinto: un comando para verificar, un atajo para la terminal del examen, un comportamiento observado en un clúster real o un riesgo de producción. Ninguna le advierte sobre preguntas capciosas, porque el CKS no las tiene: es un examen práctico que le asigna tareas y califica el estado final del clúster. Conviene reconocerlas de un vistazo, porque no todas tienen la misma urgencia.

> **Nota**
> Información complementaria que amplía lo que acaba de leer, casi siempre con la forma de comprobarlo usted mismo. Se puede saltar sin perder el hilo. Ejemplo: kube-bench no necesita instalarse en el nodo. El proyecto publica un manifiesto de Job que lo corre dentro del clúster y monta en modo de solo lectura las carpetas del nodo que revisa, como `/etc/kubernetes` y `/var/lib/kubelet`.

Consejo
Un atajo para la terminal del examen, una regla para recordar o una forma más rápida de hacer algo. Ejemplo: un tag se puede mover, un digest no. Quien tenga permiso de escritura sobre el registro puede reapuntar `v1.2` hacia otra imagen, mientras que el digest `sha256:...` identifica siempre el mismo contenido.

Advertencia
Un error frecuente o un comportamiento que sorprende en un clúster real, de esos que dejan una tarea a medio resolver. Si usted solo va a leer un tipo de caja antes del examen, que sean estas. Ejemplo: RBAC no tiene reglas de denegar. Los permisos son puramente aditivos, así que si alguien tiene dos Roles, puede hacer la suma de lo que permiten los dos, y la única forma de quitarle algo es quitarle el enlace que se lo da.

Precaución
Algo que puede causar daño real en un clúster de producción: pérdida de datos, caída de servicio o un hueco de seguridad. Son pocas y van en serio. Ejemplo: una política de auditoría con nivel `RequestResponse` sobre Secrets copia el contenido de cada Secret leído en el archivo de auditoría. Ese archivo pasa a ser tan sensible como etcd y necesita la misma protección.

Créditos y licencias de terceros

Este libro reproduce y traduce fragmentos de la documentación oficial de Kubernetes, publicada por The Kubernetes Authors bajo licencia Creative Commons Attribution 4.0 International (CC BY 4.0). También cita la documentación oficial de Falco, publicada bajo la misma licencia CC BY 4.0. Los cambios respecto del original son de dos clases: la traducción al español y el relato cotidiano con el que se acompaña cada concepto; la definición oficial aparece siempre aparte, con su respectivo enlace a la fuente.

Las traducciones al español son propias de este libro y no han sido revisadas ni avaladas por The Kubernetes Authors ni por los demás proyectos citados. Si una página oficial cambia después de la publicación del libro, la fuente vigente es la que conviene consultar.

Kubernetes es una marca registrada de The Linux Foundation. Los nombres de las certificaciones citadas pertenecen a la Linux Foundation y a la Cloud Native Computing Foundation. Este libro es material de estudio independiente y no está afiliado a esas organizaciones ni a ninguno de los proyectos o empresas mencionados, ni cuenta con su respaldo o patrocinio. Los nombres de los proyectos que aparecen en el libro, como Falco, Trivy, kube-bench, Cosign, Cilium, Istio, gVisor o kind, pertenecen a sus respectivos titulares y se mencionan solo para identificar el software con el que se practica.

Prefacio

Como esta organizado este libro, con que voz explica el examen CKS, y por que confiar por defecto es muy peligroso

Este libro nace de la misma pregunta que le dio origen a toda la serie "Lo que pasa es que": ¿y si un examen técnico se explicara con la misma voz con la que se cuenta la vida en el barrio? Al examen de seguridad de Kubernetes, el CKS (Certified Kubernetes Security Specialist), lo pintan como una lista de herramientas sueltas: RBAC, NetworkPolicy, seccomp, AppArmor, Falco, Cosign. Pero debajo de ese vocabulario, asegurar un clúster se parece mucho a cuidar un conjunto residencial: usted no construyó los edificios ni escoge quién vive en ellos, pero sí decide quién entra por la portería, qué carné lleva cada persona, qué puertas abre cada llave, qué cámaras funcionan y qué se hace la noche en que alguien entró sin permiso. Ese es el hilo de los quince capítulos.

Este es el cuarto libro de la serie, y cada uno se apoya en el anterior. El libro 1, "Lo que pasa es que Kubernetes", explica las bases. El libro 2, "Lo que pasa es que administrar Kubernetes", enseña a instalar y operar el clúster. El libro 3, "Lo que pasa es que desarrollar en Kubernetes", lleva las aplicaciones por todo su ciclo de vida. Este libro 4 agradece todo ese camino y se pone del otro lado: el de quien tiene que impedir que alguien abuse de lo que los otros tres construyeron. Cuando un tema ya se explicó antes, como RBAC en el libro 2 o las NetworkPolicies en los libros 2 y 3, aquí se recuerda en pocas líneas y se mira con ojos de atacante.

Porque esa es la postura que pide este libro: ser paranoico, a propósito. ¿Qué pasa si esta credencial se filtra? ¿Qué pasa si esta imagen trae algo que nadie revisó? ¿Qué pasa si quien dice venir de parte del kube-apiserver no viene de ahí? Nadie entra porque diga quién es: entra porque lo demostró, y cada persona o aplicación tiene permiso para hacer lo que se le encargó, ni una coma más. El carné no se presta, y

la noche de la emergencia ya debe existir un plan escrito. La seguridad se descuida porque casi nunca pasa nada, hasta el día en que pasa: ese día una empresa puede perder los datos de todos sus clientes, con sus direcciones y sus documentos, y esa confianza no vuelve fácil.

También conviene ser honestos desde la primera página. Ninguna combinación de cifrado, permisos y políticas deja un clúster invulnerable: siempre puede aparecer una vulnerabilidad nueva, o alguien puede robar las credenciales de un administrador. Lo que sí hacen esas capas es que cada paso del atacante cueste más, se note antes y quede registrado. Y ese costo importa cada vez más. En noviembre de 2025, Anthropic documentó una campaña de espionaje en la que la IA ejecutó entre el 80 y el 90 por ciento del trabajo, contra unos treinta objetivos, y la describió como el primer caso documentado de un ciberataque a gran escala sin intervención humana sustancial. En febrero de 2026, el informe anual de CrowdStrike reportó que las operaciones de adversarios que usan IA crecieron un 89 por ciento en un año, según la medición de esa empresa. Cuando el atacante trabaja a la velocidad de una máquina, la seguridad deja de ser un pendiente para después.

El temario oficial del CKS, publicado por la CNCF (Cloud Native Computing Foundation), describe lo que sabe hacer quien asegura un clúster:

- Restringir el acceso dentro del clúster con políticas de red.
- Revisar la configuración de los componentes contra el CIS Benchmark (Center for Internet Security).
- Configurar Ingress con TLS (Transport Layer Security), proteger los metadatos y los endpoints de los nodos, y verificar los binarios de la plataforma antes de desplegarlos.
- Reducir lo que exponen RBAC (Role-Based Access Control) y las ServiceAccounts, restringir el acceso a la API y actualizar Kubernetes para cerrar vulnerabilidades.
- Reducir la superficie del sistema operativo del nodo, aplicar el mínimo privilegio y usar herramientas de endurecimiento del kernel como AppArmor y seccomp.
- Aplicar los Pod Security Standards, manejar los Secrets, aislar cargas con contenedores en sandbox y cifrar el tráfico entre Pods con Cilium o Istio.
- Reducir la imagen base, entender la cadena de suministro con un SBOM (Soft-

ware Bill of Materials), firmar y validar artefactos, y analizar manifiestos e imágenes con herramientas como kubesec y KubeLinter.
- Detectar comportamiento malicioso, investigar las fases de un ataque, mantener los contenedores inmutables y leer los logs de auditoría.

De qué se trata este libro

Este es un libro de preparación para el examen de certificación CKS, de la CNCF y la Linux Foundation. Sigue el temario oficial v1.34, organizado en seis partes que siguen los seis dominios del examen: Configuración del clúster (15%), Fortalecimiento del clúster (15%), Fortalecimiento del sistema (10%), Minimizar las vulnerabilidades de los microservicios (20%), Seguridad de la cadena de suministro (20%) y Monitoreo, registro y seguridad en tiempo de ejecución (20%). Son quince capítulos, cada uno abierto con una escena del conjunto Los Nogales que se convierte en la explicación técnica de un mecanismo real de seguridad, y cerrado con un resumen que recoge esa escena. Cada parte tiene su propio laboratorio sobre un clúster kind real.

Un aviso sobre la versión. Al escribir esta edición, el temario publicado por la CNCF es el v1.34, y la página oficial de la certificación indica que el examen se basa en Kubernetes v1.35. Por eso los ejemplos y los laboratorios de este libro corren sobre Kubernetes v1.35. Antes de agendar su examen, revise esa página y el repositorio github.com/cncf/curriculum: si el temario cambió, esa página manda sobre este libro.

De qué no se trata este libro

Este libro no es un manual de referencia de cada campo, cada bandera ni cada herramienta que menciona; para eso está la documentación oficial de cada proyecto, que se cita en cada capítulo. Tampoco es un curso de Kubernetes desde cero: la página oficial del CKS exige haber presentado y aprobado el CKA antes de intentar este examen, así que el libro asume que usted ya administra un clúster, ya sabe qué son un Deployment y un Service, y ya resolvió tareas en una terminal contra el reloj. Ningún proveedor de nube recibe un capítulo propio: el libro se queda en lo que el examen evalúa sobre un clúster genérico. Y no reproduce preguntas de exámenes reales ni de simuladores como killer.sh: eso violaría la ética y el acuerdo de confidencialidad del

examen, y además no le serviría, porque la tarea del examen llega con otro namespace, otra imagen y otra política.

Si alguno de los temas de base le queda grande, los libros anteriores de la serie los explican con calma: los conceptos de Kubernetes en el libro 1, la operación del clúster en el libro 2 y el ciclo de vida de las aplicaciones en el libro 3.

Cómo está organizado

El libro sigue el orden de los seis dominios del examen, agrupados en seis partes, y cuenta una sola historia: la de Los Nogales, desde la noche en que abre sin reglamento hasta la mañana en que puede reconstruir, minuto a minuto, quién entró y qué hizo. En la primera parte (configuración del clúster), el conjunto escribe el reglamento de quién entra a cada torre con NetworkPolicy y lo compara con el CIS Benchmark usando kube-bench (capítulo 1), y luego instala una portería que verifica la cédula, con TLS en el Ingress y binarios verificados antes de desplegarlos (capítulo 2). En la segunda parte (fortalecimiento del clúster), cada quien recibe un carné que abre solo sus puertas, con RBAC y ServiceAccounts de mínimo privilegio (capítulo 3), y la reja eléctrica de la administración, la API del clúster, se cierra a extraños y se mantiene al día con cada versión (capítulo 4). En la tercera parte (fortalecimiento del sistema), se vacían las piezas que nadie usa en el sistema operativo del nodo (capítulo 5) y se cierran las salidas traseras con AppArmor y seccomp (capítulo 6). La cuarta parte (vulnerabilidades de los microservicios) pone el reglamento de convivencia con los Pod Security Standards (capítulo 7), revisa la caja fuerte de los Secrets y el cifrado de etcd (capítulo 8), y levanta paredes propias con sandboxes y un citófono cifrado entre Pods (capítulo 9). La quinta parte (cadena de suministro) recorre la tienda del conjunto: la vitrina con solo lo que se vende, es decir, la imagen base mínima (capítulo 10), la factura de dónde vino cada caja con un SBOM (capítulo 11) y el proveedor autorizado con su sello de garantía, firmas y análisis estático (capítulo 12). La sexta parte (monitoreo y tiempo de ejecución) instala las cámaras que aprenden lo normal con Falco (capítulo 13), reconstruye la película de quién entró con los logs de auditoría (capítulo 14) y pone el candado que no se abre a mitad de turno, con contenedores inmutables (capítulo 15).

Para que la historia no se enrede, la analogía base es fija en todo el libro:

Kubernetes	En el conjunto Los Nogales
Clúster	El conjunto residencial completo
Namespace	Una torre del conjunto
Pod	Un apartamento, con quienes viven en él
Nodo	El terreno y la estructura donde se levantan los apartamentos
kube-apiserver	La administración, a la que se llega por la reja eléctrica
Usuario y ServiceAccount	Un residente con su carné, y un robot de servicio con el suyo
NetworkPolicy	El reglamento de quién entra a cada torre
Admisión y Pod Security	El reglamento de convivencia que se revisa en la portería

La seguridad del conjunto la lleva Justo Segura, el celador que usted ya conoce si leyó el libro 2. Dentro de cada parte, los capítulos avanzan de lo más general a lo más específico, y cada uno arranca recordando lo que dejaron los anteriores. El glosario al final reúne los términos clave con esta misma analogía, y el laboratorio de cada parte asume que usted ya leyó los capítulos de esa parte.

Sobre los comandos y ejemplos de código

Todos los ejemplos de YAML, de comandos y de configuración de herramientas de este libro se ejecutaron en un clúster kind con Kubernetes v1.35, y la salida que aparece impresa es la que devolvió esa ejecución. Cada herramienta se verificó contra su documentación oficial: kubernetes.io para Kubernetes, falco.org para Falco, docs.sigstore.dev para Cosign y la de cada proyecto para Trivy, kube-bench, Cilium e Istio. El libro le enseña a usar esas herramientas, pero no copia sus pasos de instalación: cada capítulo le deja el enlace a la página oficial donde se explica cómo instalarlas,

porque esos pasos cambian más rápido que una edición impresa. Cuando un comportamiento depende de la versión, el libro lo dice y le muestra cómo consultarla.

Todo el código de los laboratorios está publicado en https://github.com/lfarizav/lo-que-pasa-es-que-asegurar-kubernetes. El repositorio tiene dos carpetas: `es/` para la edición en español y `en/` para la edición en inglés. Dentro de cada una hay un laboratorio por parte, de `lab-part-01` a `lab-part-06`, con sus comandos, sus comprobaciones y las versiones con las que se probó, listo para clonar y ejecutar en una terminal de Linux:

```
git clone https://github.com/lfarizav/lo-que-pasa-\
es-que-asegurar-kubernetes.git
cd lo-que-pasa-es-que-asegurar-kubernetes/es
```

Cada laboratorio levanta su propio clúster kind aislado y escribe su kubeconfig en un archivo privado, sin tocar su kubeconfig de siempre ni otros clústeres que usted tenga. Nunca lo corra contra un clúster de producción: varios pasos cambian a propósito la configuración del plano de control o cargan perfiles en el kernel del equipo. Cada paso le dice el objetivo y el entregable, y la secuencia de comandos queda guardada aparte para que usted la revele después de intentarlo por su cuenta. La calificación consulta el clúster en vivo y no lee su historial de comandos. Se parecen a lo que le van a pedir, porque el CKS no es de opción múltiple: según su página oficial, es una prueba práctica de dos horas en la línea de comandos. Aun así no son un simulador ni reproducen preguntas del examen.

Cada capítulo cierra, justo antes de su resumen, con una sección **Practique en el laboratorio** que le dice qué laboratorio le corresponde y qué temas va a trabajar allí; los pasos, los comandos y las comprobaciones están solo en el repositorio.

Cómo contactarnos

Este libro es parte del catálogo de Luis Felipe Ariza Vesga. Si encuentra un ejemplo que no se comporta como el libro dice, un campo que cambió de nombre en una versión reciente de Kubernetes, una regla de Falco que ya no coincide con la versión publicada, un comando de Cosign que cambió, o una afirmación que no coincide con la documentación oficial vigente, esa retroalimentación es bienvenida y se toma en

serio. En seguridad, una afirmación desactualizada no es un detalle menor: es una vulnerabilidad documentada. Kubernetes y las herramientas que lo rodean publican versiones nuevas cada pocos meses, así que ninguna edición es la última palabra, y la corrección hecha a tiempo vale más que la promesa de que nunca hará falta. Este libro prefiere quedar corregido a quedar bonito, igual que un buen equipo de seguridad prefiere una alerta que suene a tiempo, aunque sea falsa, a un silencio tranquilo que solo significa que nadie estaba mirando.

Agradecimientos

Este libro no inventó nada desde cero, y eso es justamente lo que quiere agradecer. Kubernetes y su seguridad existen porque miles de personas decidieron construir las mismas herramientas entre todas, a la vista de todos, en lugar de guardarlas cada una para sí.

A la Cloud Native Computing Foundation (CNCF) y a la Linux Foundation, por darle una casa neutral a Kubernetes y a los proyectos que lo rodean, por publicar abierto el temario del examen CKS y por sostener la certificación que este libro prepara.

A las comunidades y a las empresas que donaron proyectos completos a la CNCF. Falco, Cilium, Istio, Sigstore y el propio Kubernetes llegaron así a cualquier administrador del mundo, sin pedirle permiso a nadie. Cada capítulo de este libro se apoya en una de esas donaciones.

A los mantenedores de Kubernetes, etcd, Falco, Trivy, kube-bench, kubesec, KubeLinter, Sigstore, Cilium, Istio, gVisor y kind. Responder reportes de seguridad a medianoche, publicar un parche antes de que la vulnerabilidad se vuelva noticia y decir que no a una función mal pensada es un trabajo que casi nadie ve y del que todo el mundo depende.

A todos los contribuidores: a quienes escriben código, pero también a quienes escriben y traducen la documentación, reportan errores, prueban versiones beta y responden preguntas en los foros. La documentación que este libro cita en cada capítulo existe gracias a ellos, abierta y gratis para cualquiera.

A las comunidades locales de código abierto: a los miembros de Cloud Native Co-

lombia, de KCD Colombia (Kubernetes Community Days) y del Colombia OpenInfra User Group. Organizan meetups, talleres y eventos para fomentar el uso del código abierto sin cobrar por ello, y muchas veces ponen de su propio bolsillo el refrigerio, el transporte o el alquiler del lugar para que cada encuentro se haga realidad. Buena parte de lo que aquí se cuenta se aprendió en esas reuniones.

Y gracias, por último, al warrah de las Malvinas, cuya historia le presta a la portada su lección más dura: un animal que nunca tuvo motivo para desconfiar no alcanzó a aprenderlo el día en que por fin le hizo falta.

PARTE I

Configuración del clúster (15%)

1

El reglamento de quién entra a cada torre

Por qué un clúster sin NetworkPolicy deja que cualquier torre hable con cualquier otra, por qué hacen falta dos políticas y no una, y qué revisa el CIS Benchmark antes de que usted confíe en el resultado.

1.1 El conjunto sin reglamento de visitas

El conjunto Los Nogales recién terminó de construirse. Ocho torres, un parque central, una piscina que todavía huele a cloro nuevo y una portería con un solo celador en el turno de la mañana: Justo Segura, el mismo que en el libro 2 cuidaba la finca y decidía qué tráfico pasaba de una casa a otra, y que ahora estrena trabajo en un conjunto residencial. Sobre el mostrador de la portería, junto al libro de visitas todavía sin estrenar, Justo acomodó desde el primer día una figurita de madera: un animalito flaco como un perro pequeño, de patas cortas y cola terminada en punta blanca. Nadie le preguntó de dónde sacó esa figura; sin embargo, un día dijo con voz seria: recuerde que confianza no es lo mismo que seguridad.

El problema es que todavía no se ha reunido la asamblea de copropietarios, no hay reglamento firmado y nadie le entregó a Justo una lista de quién vive en cuál torre ni de quién puede visitar a quién. Así que hace lo único que puede hacer sin instrucciones: deja pasar a todo el mundo. Un visitante dice que va para la torre 3 y entra sin que

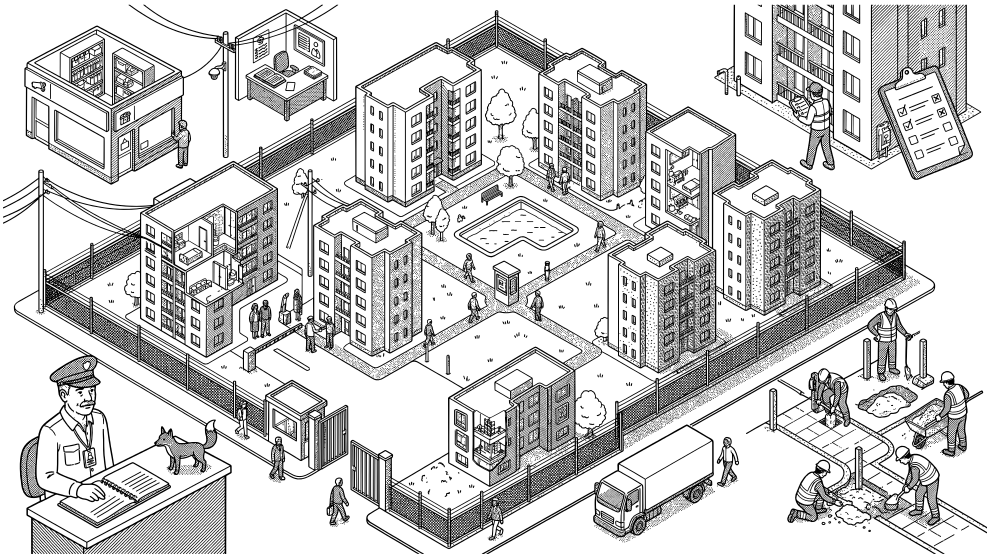


Figura 1.1. El conjunto Los Nogales completo, el mapa de este libro: la portería con Justo Segura y su figurita, las ocho torres alrededor del parque, la reja eléctrica, la tienda de la esquina, la administración y, abajo a la derecha, los trabajadores que ponen los postes de los senderos. Cada capítulo baja a una parte de este mapa.

nadie lo verifique. Otro dice que va para la torre 7, y también entra. El camión de gas llega directo hasta el parqueadero de la torre 1, aunque nadie de esa torre pidió gas ese día. No es que el celador sea negligente: es que, sin reglamento, no tiene ninguna base para negarle el paso a nadie, y decir que sí a todo es, literalmente, lo único que le toca hacer, sin olvidar requisar a las personas a la entrada y a la salida. La Figura 1.1 muestra el conjunto completo que este libro recorre capítulo por capítulo.

Lo que pasa es que un clúster de Kubernetes recién creado se comporta igual, y da miedo una vez se compara con la analogía del conjunto. La página oficial sobre Network Policies lo explica así: por defecto, un Pod no está aislado ni para el tráfico que entra ni para el que sale, y todas las conexiones están permitidas. Un Pod del namespace de facturación puede hablar con la base de datos de otro equipo, con el panel de administración o con cualquier otro Pod del clúster, simplemente porque nadie escribió todavía el reglamento. Por eso el temario del CKS abre su primer dominio, Configuración del clúster, con esta tarea: usar políticas de red para restringir el acceso

dentro del clúster.

Póngase un momento en los zapatos de quien quiere entrar. Esa persona no necesita la llave de la portería: le basta con controlar un solo Pod, por ejemplo aprovechando una vulnerabilidad de una aplicación web expuesta. Desde ahí, en un clúster sin reglamento, puede recorrer el conjunto torre por torre, probar qué servicios contestan y llegar a lo que de verdad le interesa, que casi nunca es el Pod por el que entró. Un reglamento de red no impide que la persona entre a ese primer apartamento, pero sí decide si desde ahí puede caminar al resto del conjunto. En el libro 3, en el capítulo sobre con quién se puede hablar desde el local, ya vio cómo se escribe una NetworkPolicy desde el lado de la aplicación; aquí se mira desde el lado de quien la tiene que hacer cumplir.

1.2 Dos reglamentos, no uno

El reglamento se escribe con un objeto que Kubernetes llama NetworkPolicy. Vive dentro de un namespace, igual que el reglamento de una torre solo aplica a esa torre. Con `podSelector` escoge a qué Pods gobierna, y con `policyTypes` declara si regula el tráfico que entra (`Ingress`), el que sale (`Egress`) o ambos. Basta con que una política seleccione a un Pod para que ese Pod quede aislado en las direcciones que la política declara: a partir de ese momento solo pasa lo que alguna política permita de forma explícita. La Figura 1.2 muestra la portería antes del reglamento, y la Figura 1.3, las dos políticas entre torres: la barrera para todos y el carril para lo autorizado.

Para ver la diferencia hace falta un conjunto de prueba. El archivo `torres.yaml` crea dos namespaces, `torre-3` y `torre-7`; en la torre 3 vive `residente`, un servidor web con su Service, y en la torre 7 hay dos visitantes: `visita-autorizada`, con la etiqueta `rol: visita-autorizada`, y `desconocido`, sin etiqueta. Antes de escribir cualquier reglamento, el desconocido entra sin problema:

```
kubectl -n torre-7 exec desconocido -- wget -qO- -T 3
 ↪ http://residente.torre-3 | grep -o '<title>.*</title>'
```

```
<title>Welcome to nginx!</title>
```

Un Pod cualquiera de otra torre leyó la página del residente sin que nadie le pregun-



Figura 1.2. Justo Segura en la portería, con el libro de visitas sin estrenar y la figurita del animal flaco de cola con punta blanca. Sin reglamento, deja pasar a todos: así se comporta un clúster sin NetworkPolicy.



Figura 1.3. Entre dos torres, una barrera cerrada para todos (la política que niega todo) y, al lado, un celador que deja pasar a un visitante con pase autorizado (la política que permite solo el tráfico legítimo).

tara nada. Ese es el clúster recién creado del que habla la documentación.

Aquí viene el punto más importante del capítulo: asegurar una torre pide **dos** NetworkPolicies, no una. La primera cierra todo. Es el reglamento que la documentación llama "default deny": un `podSelector` vacío, que selecciona todos los Pods del namespace, con `Ingress` y `Egress` en `policyTypes` y sin ninguna regla que permita algo.

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: negar-todo
  namespace: torre-3
spec:
  podSelector: {}
  policyTypes:
  - Ingress
```

```
  - Egress
```

Las llaves vacías de `podSelector: {}` significan "todos los Pods de este namespace", y la ausencia de bloques `ingress` y `egress` significa "nada permitido". Al aplicarla, la torre queda cerrada en las dos direcciones:

```
kubectl apply -f 01-negar-todo.yaml
kubectl -n torre-7 exec visita-autorizada -- wget -qO- -T
 ↪  3 http://residente.torre-3
kubectl -n torre-3 exec residente -- nslookup -timeout=3
 ↪  kubernetes.default.svc.cluster.local
```

```
networkpolicy.networking.k8s.io/negar-todo created
wget: download timed out
command terminated with exit code 1
;; connection timed out; no servers could be reached
command terminated with exit code 1
```

Ya no entra nadie, ni siquiera la visita autorizada. Y hay un efecto que sorprende la primera vez: el propio residente perdió el DNS (Domain Name System), porque preguntarle a CoreDNS por un nombre también es tráfico de salida. La documentación lo advierte con estas palabras: "una política de egress que niega todo también bloquea el tráfico de DNS. Si sus cargas necesitan resolver nombres, debe agregar una NetworkPolicy aparte que permita el egress hacia el servicio de DNS del clúster".

Por eso existe la segunda política. Sobre la base cerrada, abre solo lo necesario: la salida hacia los Pods de CoreDNS, que llevan la etiqueta `k8s-app: kube-dns` en el namespace `kube-system`, por el puerto 53 en UDP (User Datagram Protocol) y TCP (Transmission Control Protocol), y la entrada al puerto 80 solo para los Pods con `rol: visita-autorizada` que vienen de la torre 7.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: permitir-lo-justo
```

```
  namespace: torre-3
spec:
  podSelector:
    matchLabels:
      app: residente
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: torre-7
      podSelector:
        matchLabels:
          rol: visita-autorizada
    ports:
    - protocol: TCP
      port: 80
  egress:
  - to:
    - namespaceSelector:
        matchLabels:
          kubernetes.io/metadata.name: kube-system
      podSelector:
        matchLabels:
          k8s-app: kube-dns
    ports:
    - protocol: UDP
      port: 53
    - protocol: TCP
      port: 53
```

Fíjese en un detalle de forma: en `from` y en `to`, el `namespaceSelector` y el `podSelector` están dentro del mismo elemento de la lista, sin un guion entre ellos. Eso significa que el origen tiene que cumplir las dos condiciones a la vez: estar en la torre 7 **y** llevar la etiqueta de visita autorizada. Con la segunda política aplicada, cada visitante recibe la respuesta que le corresponde:

```
kubectl apply -f 02-permitir-lo-justo.yaml
kubectl -n torre-3 get networkpolicy
kubectl -n torre-7 exec visita-autorizada -- wget -qO- -T
 ↪  3 http://residente.torre-3 | grep -o
 ↪  '<title>.*</title>'
kubectl -n torre-7 exec desconocido -- wget -qO- -T 3
 ↪  http://residente.torre-3
kubectl -n torre-3 exec residente -- nslookup
 ↪  kubernetes.default.svc.cluster.local
```

```
networkpolicy.networking.k8s.io/permitir-lo-justo created
NAME                POD-SELECTOR    AGE
negar-todo          <none>          9s
permitir-lo-justo   app=residente   0s
<title>Welcome to nginx!</title>
wget: download timed out
command terminated with exit code 1
Server:     10.96.0.10
Address:    10.96.0.10:53
Name:   kubernetes.default.svc.cluster.local
Address: 10.96.0.1
```

La visita autorizada entra, el desconocido se queda en la puerta y el residente vuelve a resolver nombres con el servicio de DNS del clúster, que responde desde la dirección 10.96.0.10. Las dos políticas conviven sin pelear, y la razón la da la documentación: "las políticas de red no entran en conflicto; son aditivas. Si una o varias políticas se aplican a un Pod en una dirección, las conexiones permitidas en esa dirección son la unión de lo que permiten las políticas aplicables". Cada política que selecciona al Pod agrega permisos y ninguna los quita; el orden en que se crearon no cambia el resultado. Por eso el patrón es siempre el mismo: una política que niega todo y, encima, las que abren lo justo.

Advertencia
Un guion de más cambia el significado de la política. Si `namespaceSelector` y `podSelector` van en el mismo elemento de `from`, el origen debe cumplir las dos condiciones. Si cada uno empieza con su propio guion, son dos elementos y basta con cumplir una: entraría cualquier Pod de la torre 7, o cualquier Pod con la etiqueta en la propia torre. La documentación recomienda revisar con `kubectl describe networkpolicy` cómo interpretó Kubernetes la política antes de darla por buena.

Precaución
Para seleccionar un namespace, use la etiqueta `kubernetes.io/metadata.name`. Según la documentación, el plano de control la pone en todos los namespaces, con el nombre como valor, y no se puede cambiar. Una etiqueta inventada, como `torre: siete`, la puede copiar cualquiera que tenga permiso para etiquetar namespaces, y con eso su namespace pasaría a cumplir la regla de entrada. Es la versión en Kubernetes de quien se pone el uniforme de otro para que lo dejen pasar.

1.3 La condición que decide si el reglamento se cumple

Hay una condición que ninguna de las dos políticas menciona, y es la que decide si el reglamento se cumple o se queda pegado en la cartelera sin que nadie lo lea: que el clúster tenga quién lo haga cumplir. Kubernetes guarda la NetworkPolicy, pero no la aplica por sí mismo. La documentación lo dice sin dejar dudas: "las políticas de red las implementa el plugin de red. Para usarlas, usted debe tener una solución de red que soporte NetworkPolicy. Crear un recurso NetworkPolicy sin un controlador que lo implemente no tendrá ningún efecto". La Figura 1.4 lo muestra en el conjunto: sin postes, la barrera es solo una raya pintada.

En el conjunto, eso equivale a imprimir el reglamento, firmarlo en asamblea y pegarlo en la cartelera de la portería, mientras el celador de turno no sabe leer. El papel existe, `kubectl get networkpolicy` lo muestra con su nombre y su edad, y aun así ningún paquete se bloquea. Por eso los ejemplos anteriores valen como prueba: el `download timed out` del desconocido demuestra que en ese clúster sí hay un celador. El clúster de este libro usa kind, cuyo plugin de red, kindnet, aplica Network Policy desde la versión 0.24.0 de kind, según sus notas de lanzamiento. En un clúster

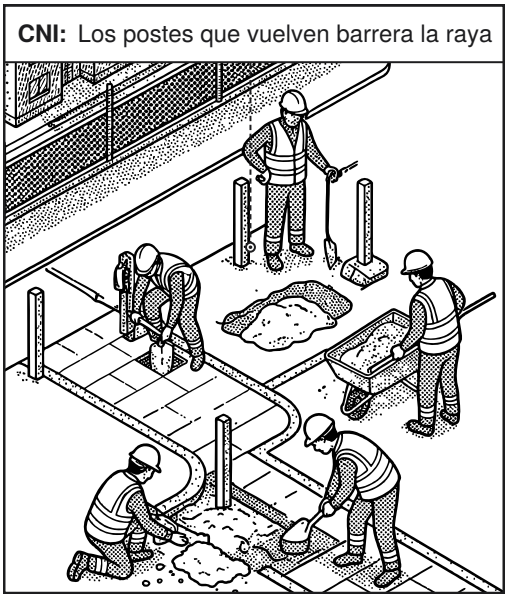


Figura 1.4. Trabajadores clavando los postes en los senderos. Sin ellos, la barrera sería una raya pintada: el plugin de red (CNI) es el que hace cumplir las NetworkPolicy.

de producción, el CNI (Container Network Interface) puede ser Calico, Cilium u otro, y la pregunta se responde igual: con una prueba, no con la fe en que el objeto aplicado está funcionando.

Esa prueba también es el primer gesto de un buen administrador de seguridad, que nunca da por hecho que algo funciona porque alguien dijo que funciona. El atacante tampoco lo da por hecho: lo primero que hará desde el Pod que controla es intentar conectarse a otros servicios, y lo único que importa es qué le responde el clúster.

> **Consejo**
> Después de aplicar las dos políticas, pruébelas antes de darlas por buenas. Desde un Pod que no debería entrar, `wget -qO- -T 3 http://<servicio>.<namespace>` tiene que terminar en `download timed out`. Si en cambio aparece la página, el CNI de ese clúster no está aplicando NetworkPolicy y el reglamento solo existe en el papel. El control 5.3.1 de kube-bench hace la misma pregunta y la deja como manual, porque no tiene forma de responderla por usted.

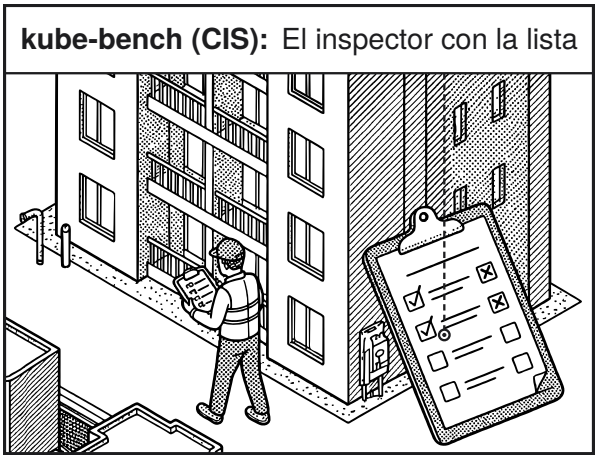


Figura 1.5. Un inspector recorre una torre con la lista de chequeo y marca cada punto: es la revisión de kube-bench contra el CIS Kubernetes Benchmark.

1.4 La revisión antes del turno

Aun con reglamento y con celador, queda un problema práctico: ¿cómo sabe la administración que cada torre tiene su reglamento colgado, y no solo la torre 3 mientras las otras siete siguen abiertas? Un namespace nuevo nace sin ninguna NetworkPolicy, igual que una torre nueva se entrega sin reglamento hasta que alguien lo escribe. La primera revisión es tan simple como `kubectl get networkpolicy --all-namespaces`, comparada contra la lista de namespaces que deberían estar aislados, y se repite antes de cada turno, porque los namespaces se crean todos los días. La Figura 1.5 muestra esa ronda.

La segunda revisión usa un estándar externo: el CIS Kubernetes Benchmark, publicado por el Center for Internet Security, que es la lista de configuraciones seguras contra la que se audita un clúster. Una herramienta que lo aplica es kube-bench, y su página de instalación explica las formas de ponerla a correr. La más cómoda es la que describe su guía de ejecución: un Job de Kubernetes que monta en modo de solo lectura las carpetas del nodo que revisa. El archivo `kube-bench-politicas.yaml` es el `job.yaml` del proyecto, en la versión v0.16.0, con un único cambio: los argumentos `run --targets policies`, para que revise solo la sección de políticas.

```
kubectl apply -f kube-bench-politicas.yaml
kubectl wait --for=condition=complete
 ↪  job/kube-bench-politicas --timeout=120s
kubectl logs job/kube-bench-politicas | grep -E
 ↪  '^\[(PASS|FAIL|WARN)\] 5\.3'
```

```
job.batch/kube-bench-politicas created
job.batch/kube-bench-politicas condition met
[WARN] 5.3.1 Ensure that the CNI in use supports
 ↪  NetworkPolicies (Manual)
[WARN] 5.3.2 Ensure that all Namespaces have
 ↪  NetworkPolicies defined (Manual)
```

Los dos controles de la sección 5.3 piden exactamente lo que este capítulo hizo: el 5.3.1, que el CNI aplique NetworkPolicy, y el 5.3.2, que todos los namespaces tengan sus políticas. Y los dos salen como advertencia y marcados como manuales. kube-bench puede leer los archivos y los argumentos de los componentes, pero no puede comprobar por usted si un paquete se bloquea ni si el reglamento de cada torre dice algo sensato. Un inspector automático que solo cuenta papeles pegados en la puerta no reemplaza al celador que lee el reglamento línea por línea, y de esa diferencia trata buena parte del próximo capítulo.

> **Nota**
> En el mismo informe, kube-bench terminó la sección de políticas con 0 controles aprobados, 0 fallidos y 34 advertencias. No es que el clúster esté bien ni mal en esos puntos: todos los controles de esa sección son manuales. Un informe sin rojos en esa parte no prueba nada; solo le dice qué le toca revisar a usted.

1.5 Practique en el laboratorio

Temas del laboratorio `lab-part-01` (Parte I) para este capítulo:

- Medir que, sin políticas, un Pod sin permiso llega al servicio de la torre.
- Escribir la política que niega todo el tráfico de entrada y de salida del namespace.

- Abrir solo la salida al DNS del clúster y la entrada del vecino autorizado, por etiqueta.
- Probar el reglamento desde el vecino autorizado y desde el desconocido, por nombre y por IP.
- Correr kube-bench como Job, separar los FAIL de los WARN, corregir un control en el manifiesto estático y volver a correrlo.

1.6 Resumen

Volvamos a Los Nogales el día en que por fin llega el reglamento. Justo Segura, que antes dejaba pasar a cualquiera porque no tenía de dónde sacar un no, ahora tiene instrucciones claras, y se aplican en este orden, el mismo recorrido de la Figura 1.6:

1. Por defecto, nadie entra ni sale de la torre: una NetworkPolicy con `podSelector: {}` e `Ingress` y `Egress` en `policyTypes`, sin reglas.
2. Encima de esa base cerrada, una segunda política abre lo justo: la salida hacia el DNS del clúster por el puerto 53 y la entrada solo para las visitas autorizadas, por namespace y por etiqueta.
3. Las políticas se suman y nunca se restan, así que el orden en que se aplican no cambia el resultado.
4. Todo eso depende de que el CNI aplique NetworkPolicy, y la única forma de saberlo es probarlo desde un Pod que no debería entrar.
5. kube-bench revisa el clúster contra el CIS Benchmark, pero marca como manuales justamente los dos controles de red: la última palabra la tiene una prueba hecha por usted.

Con el tráfico entre torres resuelto, queda otra pregunta, la del visitante que llega desde la calle y no desde otra torre: ¿la portería le pide la cédula antes de dejarlo pasar, o le basta con que diga un nombre? De eso trata el próximo capítulo.

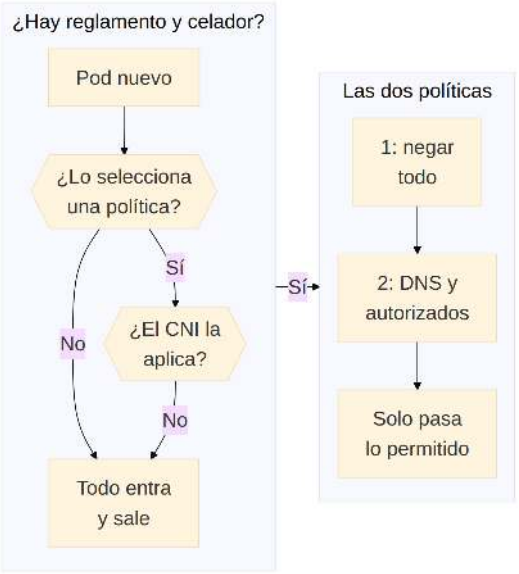


Figura 1.6. El reglamento del conjunto: sin políticas o sin un CNI que las aplique, todo entra y sale; con una política que niega todo y otra que abre lo justo, solo pasa lo permitido.

Bibliografía

- Kubernetes documentation, Network Policies https://kubernetes.io/docs/concepts/services-networking/network-policies/
- Aqua Security, kube-bench documentation, Running kube-bench https://github.com/aquasecurity/kube-bench/blob/main/docs/running.md
- CNCF, CKS Exam Curriculum v1.34 (Cluster Setup domain) https://github.com/cncf/curriculum
- kind v0.24.0 release notes (kindnet NetworkPolicy support) https://github.com/kubernetes-sigs/kind/releases/tag/v0.24.0