

# Mecanismos de Seguridad en Kubernetes



**KEEPCODING**

Tech School

# ■ Seguridad en Kubernetes

- En esta sección:
  - Introducción
  - Autenticación y Autorización / RBAC
  - Seguridad a nivel de CNI: NetworkPolicies
  - PodSecurityPolicy & PodSecurity
  - Otros sistemas y proyectos de seguridad
    - Binary Authorization
    - GKE Sandbox
    - Falco



# Introducción



# ■ Seguridad en Kubernetes

- ¿Cuándo y en qué niveles hay que preocuparse por la seguridad?
  - Siempre y en todos los niveles.
    - Código aplicación (**C**ode):
    - Imágenes docker (**C**ontainer):
      - librerías, vulnerabilidades, actualizar imágenes base
      - supply chain security (analizar todo lo que ejecutamos o sobre lo que montamos nuestras aplicaciones).
    - Kubernetes (**C**luster):
      - Proteger API Server: RBAC, quitar métodos de autenticación no seguros, etc
      - PODs: hostPath, hostNetwork, hostPort, privileged, capabilities
      - Network: políticas
      - Nodos: Vulnerabilidades Kernel, actualizaciones, detección de amenazas
    - **C**loud / Infraestructura:
      - Arquitectura segura
      - Firewalls



# ■ Seguridad en Kubernetes

- Muchas herramientas y proyectos activos
  - CIS ([Center for Internet Security](#)) Benchmarks → CIS-CAT Pro & Lite
  - Aqua Security → [kube-bench](#) (también chequea CIS)
  - Containers y CRI
    - Aqua Security → [trivy](#)
    - Secure CRIs: [gVisor](#) (Kernel Application), [Kata](#)
    - Clair, [Falco](#), [Harbor](#)
    - Google → [Binary Authorization](#)
  - Kubernetes
    - Gatekeeper, OPA
  - Networking: NetworkPolicies, WireGuard (cifrado tráfico)
  - IPS / IDS: Suricata, OSSEC

Certificación específica CKS ([Certified Kubernetes Security Specialist](#))



# API Server: Autenticación y Autorización



**KEEPCODING**

Tech School

# ■ Autenticación con el API

- Kubernetes acepta [usuarios](#) de dos tipos:
  - **ServiceAccounts** (manejadas por Kubernetes)
    - Usados por los pods para poder acceder a la API.
  - **Usuarios** normales (no manejados por Kubernetes)
    - Los usuarios normales han de autenticarse presentando un certificado en el que confíe, y Kubernetes obtendrá el usuario del "subject" del propio certificado.
- Kubernetes permite muchos tipos de [estrategias](#) de autenticación:
  - Certificados, API Tokens, Proxies externos, OIDC / SAML
  - A través del API de Kubernetes podemos hacer cualquier cosa.
  - No debemos activar HTTP basic auth ni el acceso anónimo.
- Control de identidad Cloud
  - Integrado en GKE por defecto, a través de access tokens.



# ■ Autorización con el API (RBAC)

- Kubernetes dispone de un componente de RBAC muy potente ([Role Based Access Control](#))
  - Permite seleccionar qué usuarios, grupos ó serviceaccounts pueden hacer a qué a nivel de API.
  - Por defecto en cada namespace existe una cuenta de servicio que es la que usa cada POD.
  - Podemos crear cuentas de servicio individuales y que las usen nuestros PODs.
- Nota: RBAC se aplica contra la API Kubernetes, controlamos las autorizaciones al acceder a la API de Kubernetes.





# ■ Autorización con el API (RBAC)

- Objetos RBAC
  - ServiceAccounts: Para procesos que corren en los pods (no para humanos)
  - Role y ClusterRole: Definen permisos (siempre aditivos)
    - [Role](#): relacionado con Namespace concreto
    - [ClusterRole](#): permisos a nivel de cluster, más flexibles al poder apuntar a namespaces individuales o múltiples.
  - RoleBinding y ClusterRoleBinding: Representan la otorgación de los roles a los usuarios, grupos o ServiceAccounts.
    - [Ejemplos](#)



# Autorización con el API (RBAC)

- Con RBAC podemos seleccionar qué usuarios, grupos o service accounts pueden hacer qué.
- Podemos crear cuentas de servicio individuales (ServiceAccounts), además de la cuenta por defecto que existe en cada namespace.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: ["" ] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

```
apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "jane" to read pods in the
"default" namespace.
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
- kind: User
  name: pmoncadaisla@gmail.com # case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role # this must be Role or ClusterRole
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```



# Autorización con el API (RBAC)

Permitir que un usuario sólo pueda ver “pods”, “services”, “deployments”, “configmaps” y “logs” de un namespace llamado “test-rbac”

1. Damos acceso al usuario para autenticarse.
2. Creamos el Role que define la política de acceso.
3. Asociamos mediante RoleBinding el role que hemos definido al usuario.

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: test-rbac
  name: read-only
rules:
  # The api groups that contain the resources we want to manage
  - apiGroups: [ "", "apps", "extensions", "networking.k8s.io" ]
  # The resources to which this role grants permissions
  resources: [ "pods", "pods/log", "services", "deployments",
    "configmaps", "ingress" ]
  # The permissions granted by this role
  verbs: [ "get", "list", "watch", "exec" ]
```

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: test-rbac
  name: read-only-binding
subjects:
- kind: User
  name: pmoncadaisla@gmail.com
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role #this must be Role or ClusterRole
  name: read-only
  apiGroup: rbac.authorization.k8s.io
```

## Add members, roles to "pmoncada-001" project

Enter one or more members below. Then select a role for these members to grant them access to your resources. Multiple roles allowed. [Learn more](#)

New members  
pmoncadaisla@gmail.com ✕

Role	Condition
Kubernetes Engine Cluste...	<a href="#">Add condition</a>

Get and list access to GKE Clusters.

[+ ADD ANOTHER ROLE](#)

☐ Send notification email  
This email will inform members that you've granted them access to this role for 'pmoncada-001'

SAVE

CANCEL



# API server: audit Logs



# ■ Audit logs

- La activación de los audit logs en Kubernetes son considerados una de las medidas básicas de seguridad.
  - <https://kubernetes.io/docs/tasks/debug/debug-cluster/audit/>
- Ejemplo interesante de envío y procesamiento de estos logs con el Elastic Stack [aquí](#).



# NetworkPolicies



**KEEPCODING**  
Tech School

# NetworkPolicies (conectividad dentro del cluster)

- Que tengamos las aplicaciones en Namespaces distintos no quiere decir que estén aisladas.
  - Incluso dentro del mismo Namespace no siempre queremos permitir acceso a todos los sistemas.
  - Podemos usar [NetworkPolicies](#), que **actúan como Firewall**.
  - Debemos tener un **CNI que soporte NetworkPolicies** (Calico)
  - Podemos activarlo en GKE
  - [Spec](#)
- 
- **podSelector**: Indica a qué Pods afecta la network policy
    - si está vacío afecta a todos los pods.
  - **policyTypes**: Puede ser de Ingress o Egress
  - **ingress**: desde dónde y qué puertos
  - **egress**: hacia dónde y a qué puertos.
  - **namespaceSelector / podSelector / ipBlock / ports**

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector: # A QUIEN APLICA LA POLICY
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
      ports:
      - protocol: TCP
        port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
      ports:
      - protocol: TCP
        port: 5978
```



# Ejemplos Network Policies

- Activación en GKE:

## CLÚSTER

- Automatización
- Redes
- Seguridad
- Metadatos
- Características

Cantidad máxima de pods por nodo

110

Máscara del rango de direcciones de pods por nodo: /24

Rango de direcciones del servicio

Ejemplo: 192.168.0.0/16

☐ Habilitar Dataplane V2 ?

By enabling Dataplane V2, Kubernetes network policy is also enabled.

☐ Habilitar política de red ?

☐ Habilitar la visibilidad dentro de los nodos ?





# Ejemplos Network Policies

- A los pods con label "role=db", permitir conectividad únicamente hacia o desde pods del mismo namespace.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: myproject
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          project: myproject
  egress:
  - to:
    - namespaceSelector:
        matchLabels:
          project: myproject
```

## NetworkPolicySpec:

<https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.27/#networkpolicyspec-v1-networking-k8s-io>



# PodSecurity



# ■ PodSecurity

- Es la evolución de las antiguas PodSecurityPolicies. Disponibles a partir de Kubernetes 1.23
- Permite definir políticas / niveles de seguridad para la creación de pods, permitiendo o rechazando el uso de determinados campos de su especificación.
  - [Pod Security Standards](#)
  - [Pod Security Admission Controller](#)



# PodSecurity

- Se definen 3 políticas de seguridad que se pueden aplicar
  - privileged, baseline, restricted
- El admission controller puede actuar de 3 modos diferentes:
  - enforce, audit, warn
- Los modos se aplican a las diferentes políticas, por ejemplo: enforce: baseline, audit: restricted, warn: restricted
- Se aplica a nivel de namespaces. Ejemplo [aquí](#)
- Admission Controller [configuration](#)



# Proyectos y Sistemas de Seguridad para Kubernetes



# ■ Binary authorization

- Binary authorization es un mecanismo de seguridad para controlar que sólomente se desplieguen imágenes confiables.
  - Por defecto en Kubernetes podemos desplegar imágenes de cualquier sitio
  - Tenemos que tener cuidado si un usuario despliega una imagen que puede estar comprometida.
  - Incluso si alguien consigue acceso a nuestro clúster podría desplegar imágenes que les permitiese tener una puerta trasera.
  - Podemos limitar qué imágenes se despliegan.
  - Imágenes que estén únicamente bajo nuestro control.
  - Imágenes que hayan sido construidas por nosotros y firmadas.

Laboratorio: <https://codelabs.developers.google.com/codelabs/cloud-binauthz-intro/?hl=es#0>



# ■ GKE Sandbox

- [GKE Sandbox](#) es un mecanismo de seguridad para proteger el kernel de los nodos, está basado en el proyecto [gVisor](#).
  - La seguridad dentro de contenedores Linux se ha demostrado que puede romperse.
  - Puede producirse un escalado de privilegios y acceder a recursos del cluster.
  - Puede que un contenedor conflictivo con demasiados permisos entre en conflicto con otros recursos del cluster.
  - El proyecto gVisor nos permite ejecutar los contenedores Linux en una micro-vm.  
Pretende evitar que los contenedores ejecuten código desconocido o no confiable.
  - Impacto en performance y en funcionalidad.

Laboratorio: <https://cloud.google.com/kubernetes-engine/docs/how-to/sandbox-pods>



- Falco es un proyecto OpenSource para la detección de intrusiones y anomalías de aplicaciones Cloud Native.
  - Permite detectar comportamiento fuera de lo normal dentro de aplicaciones y generar alertas por distintas vías de comunicación (Slack, Fluentd, NATS, etc).
  - Alto performance, al estar basado en eBPF (Extended Berkeley Packet Filter) y llamadas al sistema del Kernel de Linux.
    - Una shell que se ejecuta dentro de un contenedor.
    - Un proceso que crea un proceso hijo inesperado
    - Acceso a ficheros sensibles del sistema
    - Un binario estándar tipo “ls” establece una conexión de red
    - Detectar si alguien ha accedido a un secreto.
    - Detectar si se ha escrito información sensible en un ConfigMap



<https://falco.org/>





# Ejercicios



# ■ k8s security - 1

Configura un rol que únicamente permita:

- get, describe, logs, exec

Sobre (los que lo admitan):

- pods, services, deployment, configmaps



## ■ k8s security - 2

Configura una network policy que sólo permita tráfico en un namespace entre los pods del mismo namespace y hacia otro namespace específico.

Prueba la política.

