

Pilar Seguridad "Well Architected Framework" | Seguridad en EKS

Saúl Carranza
Sr. Containers Specialist SA - LATAM

Tabla de Contenido

- Introducción al AWS Well-Architected framework
- El pilar de seguridad
- ¿Cómo pueden operar mis cargas de trabajo en contenedores de forma Segura?
- Conceptos de Seguridad a Nivel Pod
- Pod Security Standards (PSS)
- Pod Security Admission (PSA)
- Policy-as-Code (PaC)
- Conclusiones



¿Por qué AWS Well-Architected Framework?



Rápida construcción y despliegue



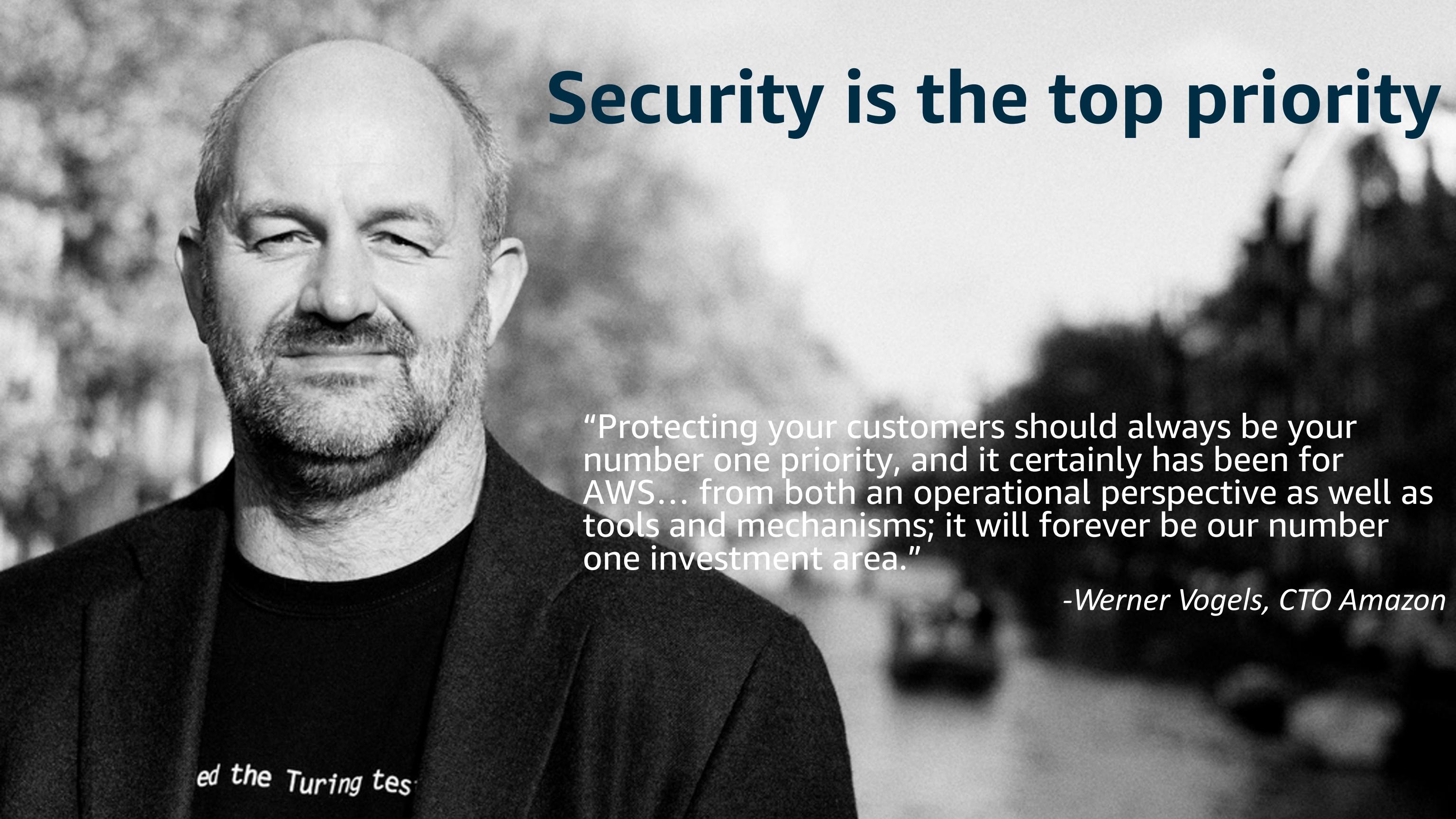
Reducir o mitigar los riesgos



Tomar decisiones basada en datos



Aprenda las mejores practicas de AWS

A black and white portrait of Werner Vogels, a middle-aged man with a beard and mustache, wearing a dark turtleneck sweater. He is smiling slightly and looking towards the camera. The background is blurred.

Security is the top priority

“Protecting your customers should always be your number one priority, and it certainly has been for AWS... from both an operational perspective as well as tools and mechanisms; it will forever be our number one investment area.”

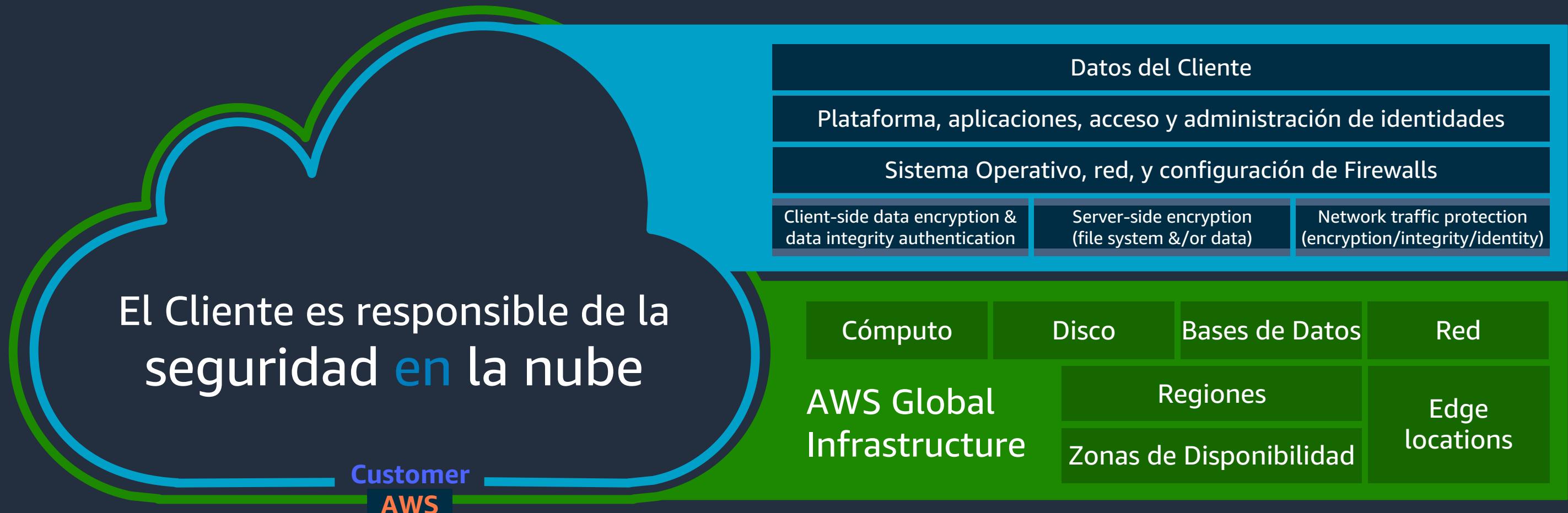
-Werner Vogels, CTO Amazon

Principios de Diseño de Seguridad

- ⊕ Implementar una base sólida de identidades
- ⊕ Habilitar trazabilidad
- ⊕ Aplicar seguridad en todas las capas
- ⊕ Automatizar buenas prácticas de seguridad
- ⊕ Proteger datos en transito y en reposo
- ⊕ Mantener a las personas lejos de los datos
- ⊕ Estar preparados para eventos de seguridad



Módulo de Responsabilidad Compartido



AWS es responsable de la seguridad **de** la nube

¿Cómo pueden operar mis cargas de trabajo en contenedores de forma Segura?

Elementos de Security Context en Kubernetes

- Puede ser aplicado a nivel pod y contenedor
- Un Security Context a nivel contenedor sobrescribe la configuración a nivel pod
- El Security Context a nivel contenedor es más granular
- Las configuraciones pueden obtenerse a través de: `kubectl explain`
`Kubectl explain pod.spec.securityContext`
`kubectl explain pod.spec.containers.securityContext`

Elementos de Security Context para Pods en Kubernetes

- Ejemplo

securityContext:

runAsUser: 1000

runAsGroup: 3000

fsGroup: 2000

Todos los contenedores del pod tendrán esta configuración, con la opción de anularla

Elementos de Security Context para Pods en Kubernetes - Ejemplo

securityContext:

allowPrivilegeEscalation: false

runAsUser: 1000

readOnlyRootFilesystem: true

runAsNonRoot: true

capabilities:

drop: ["ALL"]

seccompProfile:

type: "RuntimeDefault"

Esta es una configuración conocida por el Pod Security Standards (PSS).

Pod Security Standards (PSS) derivado de PSP

- Beta a partir de Kubernetes 1.23
- Basado en la experiencia con PSP
- Funciona con elementos de Security Context (pod y container)

“La experiencia con PodSecurityPolicy concluyó que la mayoría de los usuarios se preocupan por dos o tres políticas, lo que llevó a la creación de la Pod Security Standards, que definen tres políticas:”

Privileged - política sin restricciones.

Baseline - política mínimamente restrictiva, lo que permite establecerla como predeterminada en los pods.

Restricted - Política con las mejores prácticas de seguridad.

Pod Security Admission (PSA)

- Beta en Kubernetes 1.23, estable en Kubernetes 1.25
- Es configurado a través del API Server de K8s
- En caso de contar con una versión de K8s menor a la 1.23 puede ser usado Kubernetes Dynamic Admission Controller.

PSA Modos de Operación

- **enforce** – Infracciones a la política ocasionará el rechazo del pod.
- **audit** - Las infracciones de la política activarán la adición de una anotación de auditoría al evento registrado en el registro de auditoría, pero de lo contrario están permitidas.
- **warn** - Las infracciones de la política activarán una advertencia para el usuario, pero están permitidas.

Policy-as-Code (PaC) alternativa para PSA y PSS

- Algunas alternativas de PSS se encuentran listadas en la documentación: PaC a PSA/PSS
 - Kubewarden
 - Kyverno
 - OPA Gatekeeper
- La guía de mejores prácticas de EKS compara PaC a PSA/PSS como reemplazo de PSP

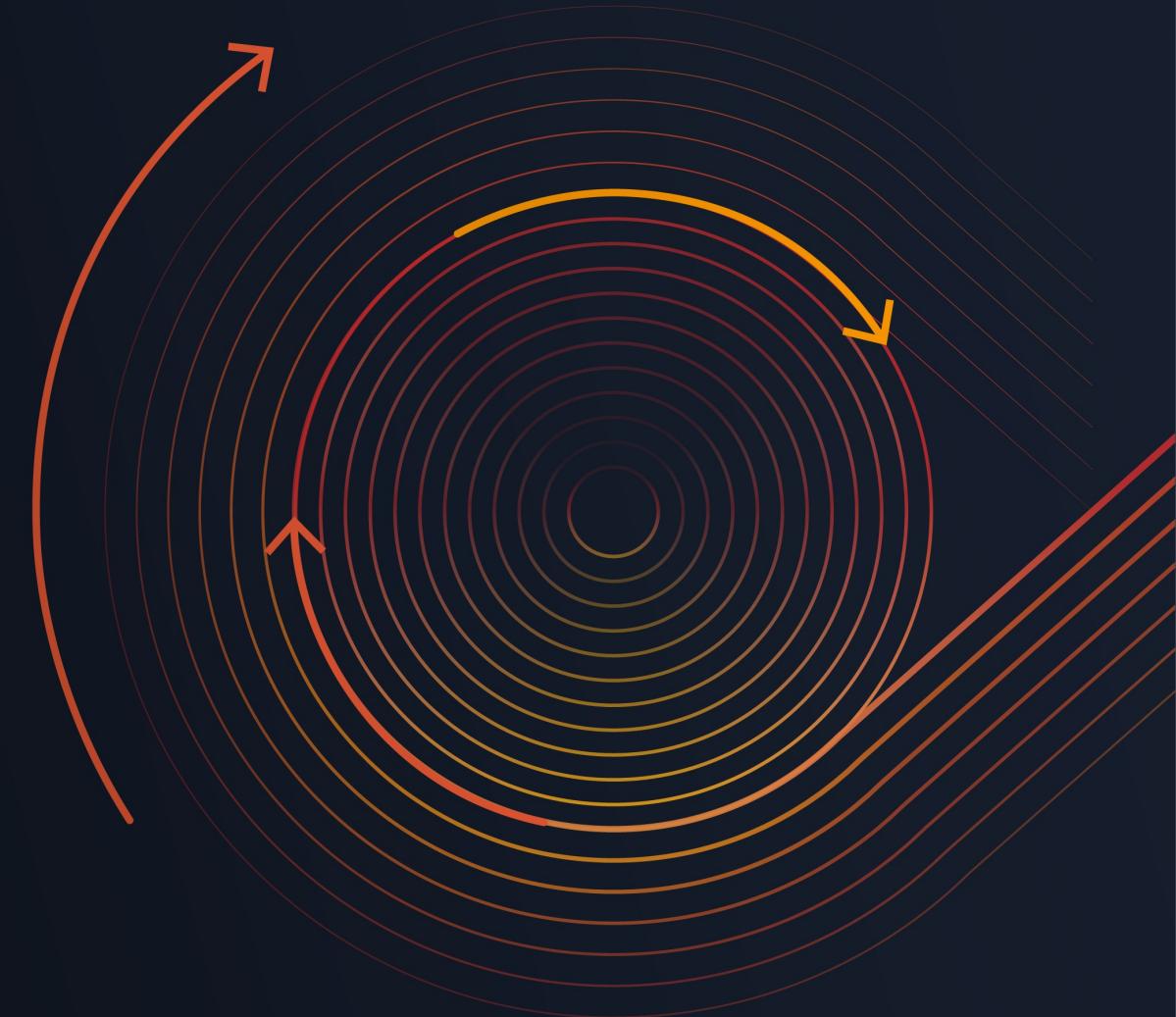
PaC no es mutuamente exclusivo de PSA/PSS

- PaC puede coexistir con PSA, PSS y PSP también.
- PaC puede aumentar el alcance de PSA y PSS al hacer cumplir las etiquetas de los Namespace con controles de mutación y validación.
- PaC también se puede usar para implementar excepciones detalladas y seguridad, por debajo del nivel del namespace.

En conclusión...

- Seguir la guia de patrones basada en el Pilar de Seguridad
- PSP quedó obsoleto en la version 1.21 de Kubernetes y eliminada en la 1.25
- PSA/PSS son la forma nática de reemplazar PSP
- PSA/PSS pueden coexistir con PSP y PaC
- PaC provee funcionalidad más allá de la incluida en PSA/PSS

Demo!!!



Gracias!!!

Saúl Carranza
✉️ ssauca@amazon.com
LinkedIn linkedin.com/in/saulcb
X [@Saul_Carranza](https://twitter.com/Saul_Carranza)

