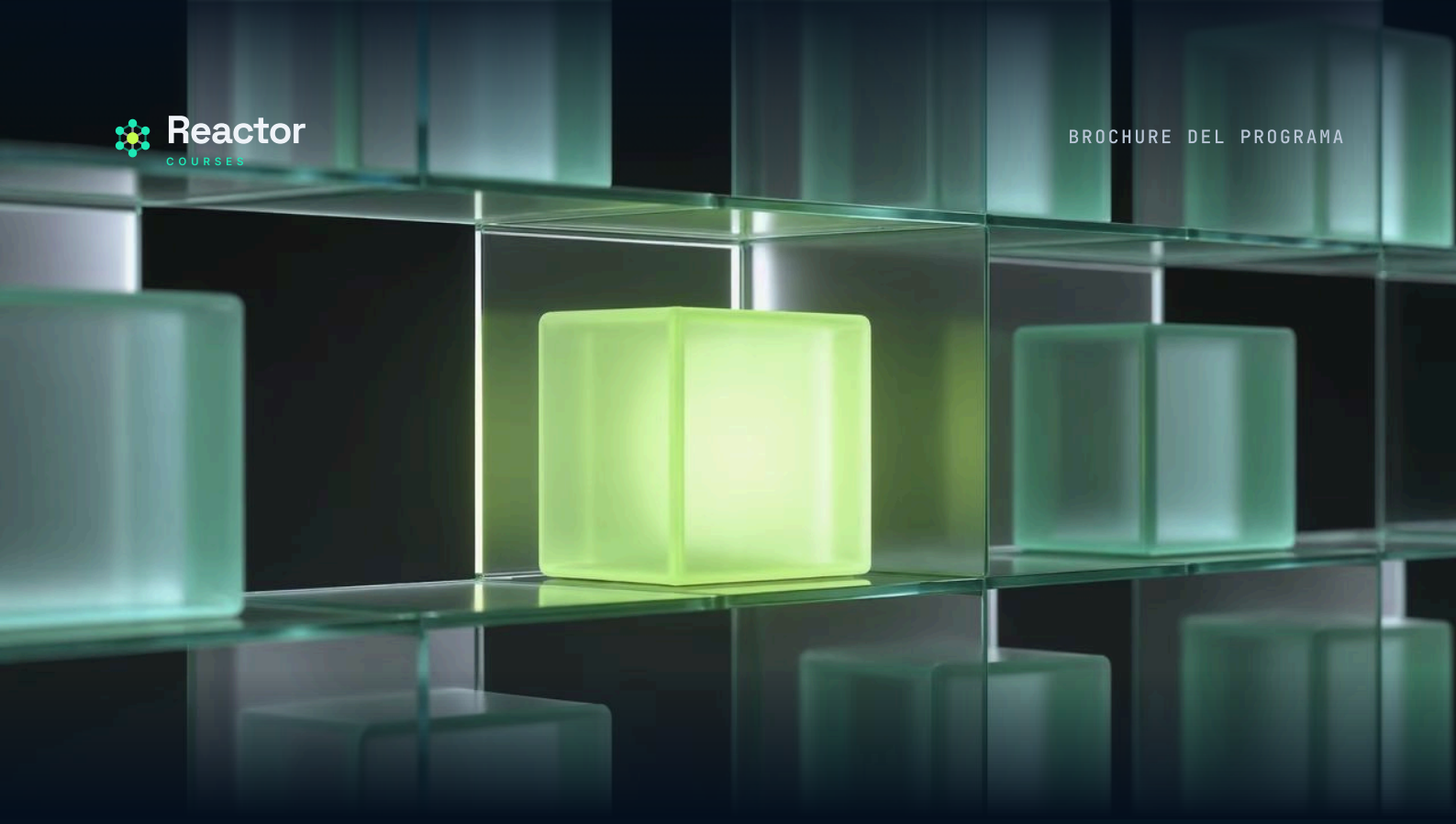
Reactor
COURSES

BROCHURE DEL PROGRAMA



ARQUITECTURA E INGENIERÍA · NUBE Y PLATAFORMAS

Kubernetes Platform Architecture

El clúster como producto compartido

Diseña, asegura, dimensiona y actualiza un clúster de Kubernetes que comparten varios equipos, con aislamiento, políticas, capacidad y ciclo de vida explícitos y probados con fallas reales.

Lo que vas a lograr Aprendes a diseñar y operar un clúster de Kubernetes que comparten varios equipos: aislado, seguro, con capacidad calculada y actualizable sin cortar el servicio.

- Tenancy y RBAC
- Políticas de admisión
- Gateway API y NetworkPolicy
- Capacidad y autoescalado
- GitOps con Argo CD
- Actualizaciones en vivo

horas en vivo	sesiones	semanas	equipos que comparten tu clúster
72	24	12	3

INICIO
Por confirmar

PRECIO
US$400 pago único · USD

MODALIDAD
Clases en vivo por Zoom · Avanzado

reactorcourses.com/programas/kubernetes-platform-architecture

EL PROGRAMA

Desplegar en Kubernetes es fácil. Que tres equipos compartan el mismo clúster sin pisarse no lo es.

Diseña, asegura, dimensiona y actualiza un clúster de Kubernetes que comparten varios equipos, con aislamiento, políticas, capacidad y ciclo de vida explícitos y probados con fallas reales. Un programa de **72 horas** para ingenieros que ya despliegan en Kubernetes y ahora tienen que administrar el clúster que usan otros equipos, sin que uno pueda tumbar o espiar al de al lado.

Duración	72 horas · 24 sesiones	Calendario	12 semanas · 2 sesiones por semana de 3 h
Modalidad	Clases en vivo por Zoom	Grabaciones	Disponibles por 12 meses
Credenciales	3 certificados con insignia de Reactor Courses	Nivel	Avanzado
Inicio	Por confirmar	Horario	Miércoles y viernes, 19:00–22:00 (GMT-5)
Precio	US$400 · pago único		

La mayoría de los cursos de Kubernetes enseñan a poner una aplicación en marcha. Este empieza donde esos terminan: cuando el clúster deja de ser tuyo y pasa a ser de todos, y cada decisión —quién ve qué, quién puede consumir cuánto, cómo se actualiza sin cortar a nadie— afecta a equipos que no controlas.

01

Aislar y gobernar

Tenancy, identidad, admisión y red

Las decisiones que separan a un equipo de otro: qué modelo de tenancy le corresponde a cada uno, qué permisos tiene, qué configuraciones se rechazan antes de llegar al clúster, qué imágenes pueden correr y con quién puede hablar cada carga. Cada control se prueba con el intento que debe impedir.

- Namespace, clúster por equipo o clúster virtual, con sus límites medidos
- RBAC, identidad de cargas y escalamiento de privilegios
- Pod Security Admission, políticas nativas y Kyverno
- NetworkPolicy, Gateway API compartido y salida controlada

02

Operar en el tiempo

Capacidad, recuperación, GitOps, costo y actualizaciones

Lo que pasa después del primer mes: cargas que se expulsan por memoria, picos que piden nodos, un namespace borrado que hay que recuperar, cambios que nadie sabe quién hizo, la factura que hay que repartir y la versión que deja de tener soporte. El programa termina con una actualización en vivo y un game day.

- Requests, QoS, expulsiones y autoescalado de pods y nodos
- Respaldo de etcd y de cargas con restauración probada
- GitOps con Argo CD y auditoría de quién cambió qué
- Costo por equipo y actualización de versión con plan de reversión

> Un namespace es una carpeta, no una pared. La pared la ponen el RBAC, las políticas de admisión, la red y las cuotas, y hay que probar cada una intentando atravesarla.

PARA QUIÉN ES

¿Es el programa indicado para ti?

Está pensado para ingenieros que ya despliegan aplicaciones en Kubernetes y quieren administrar el clúster sobre el que corren.

Es para ti si

- ✓ Ingenieros de plataforma, DevOps o SRE que ya despliegan en Kubernetes y ahora administran el clúster que usan otros equipos.
- ✓ Arquitectos que deben decidir entre clúster compartido, clúster por equipo o clústeres virtuales.
- ✓ Líderes técnicos responsables de las actualizaciones, la seguridad y el costo de clústeres existentes.
- ✓ Equipos que heredaron un clúster sin políticas ni cuotas y necesitan ordenarlo sin cortar a nadie.
- ✓ Ingenieros que tienen que migrar de Ingress NGINX a Gateway API y quieren hacerlo con criterio.

Te conviene otro programa

- — Quien nunca desplegó nada en Kubernetes: Deployment, Service e Ingress se dan por sabidos desde la primera sesión.
- — Quien busque preparar CKA, CKAD o CKS: el programa no sigue esos temarios ni prepara para esos exámenes.
- — Quien quiera aprender a programar microservicios: aquí no se diseña la aplicación, se diseña dónde corre.
- — Quien esperaba un recorrido por herramientas del ecosistema: cada herramienta entra porque resuelve una decisión, y muchas quedan fuera.
- — Quien ya cursó un programa de plataforma de nube: cerca de un tercio de este programa trata problemas parecidos — menor privilegio, identidad de cargas, segmentación, costo por equipo— a nivel del clúster. Lee en las preguntas frecuentes la que trata ese caso.
- — Quien no tenga un equipo con 16 GB de memoria o no pueda dedicar entre 10 y 12 horas por semana durante 12 semanas.

EL RESULTADO

Terminas con una plataforma para tres equipos, probada atacándola y actualizada en vivo.

Cada laboratorio deja una pieza de la plataforma de Pagos, Campañas y Catálogo, con la prueba de que hace lo que dice. Juntas forman el proyecto que defiendes en la última sesión.

01 **Respaldo de etcd probado** Snapshot del plano de control y restauración de un namespace borrado, con el tiempo de recuperación medido.

02 **Modelo de tenancy decidido** El mismo equipo como namespace y como clúster virtual, comparados con seis pruebas de escape documentadas.

03 **RBAC sin escalamiento** Permisos de tres equipos y un operador de plataforma, con diez intentos de escalamiento rechazados.

04 **Admisión que rechaza lo inseguro** Pod Security Admission, políticas nativas y Kyverno probadas en integración continua y pasadas de auditoría a imposición sin cortar cargas.

05 **Solo imágenes firmadas** Firma y SBOM con cosign y Syft, verificados en la admisión: imágenes sin firma, con otra clave o por etiqueta, rechazadas.

06 **Red aislada por equipo** Denegación por defecto, Gateway compartido donde cada equipo administra solo sus rutas y salida restringida por dominio.

07 **Recuperación de cargas con estado** Snapshot de volumen restaurado tras corromper datos, y respaldo con Velero restaurado en un clúster nuevo.

08 **Capacidad explicada con datos** Una expulsión por memoria explicada con eventos y clases QoS, y nodos dedicados que respetan los presupuestos de disrupción.

09 **Autoescalado de pods y nodos** Pico de carga absorbido por HPA y Cluster Autoscaler, con el tiempo de cada etapa medido.

10 **GitOps con auditoría** Argo CD con un proyecto por equipo que revierte cambios manuales, y un incidente reconstruido con Git y el registro de auditoría.

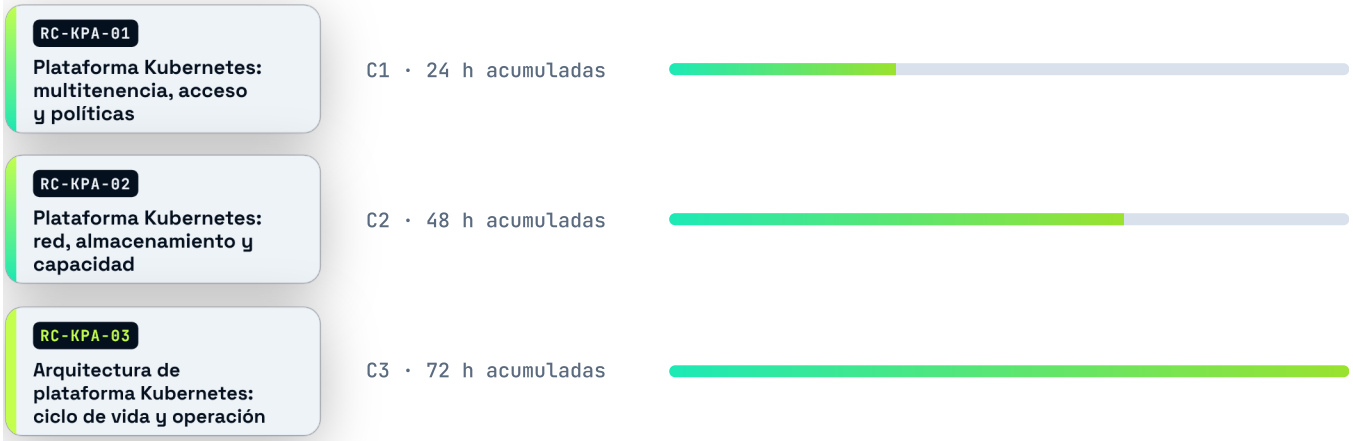
11 **Costo por equipo** Informe de OpenCost por equipo con la sobreasignación detectada y requests corregidos con evidencia.

12 **Actualización en vivo y game day** Versión menor actualizada con carga en curso sin violar presupuestos de disrupción, y dos fallas inyectadas detectadas por alertas.

RUTA DE CERTIFICACIÓN

3 credenciales progresivas, no una al final.

Cada etapa cierra un bloque completo de la plataforma: primero aislar y gobernar, después red, almacenamiento y capacidad, y al final operación y ciclo de vida. Son credenciales de Reactor Courses, no exámenes de terceros.



CERTIFICACIÓN INTEGRAL

Kubernetes Platform Architecture — El clúster como producto compartido (72 horas)

Requiere entregar la plataforma de los tres equipos, el resultado del game day y la actualización en vivo con su plan de reversión. Los certificados son emitidos por Reactor Courses y acreditan las horas y competencias del programa.

Cada certificado de Reactor Courses viene con su insignia, aparte de cualquier certificación oficial.

Cualquiera puede comprobar un certificado de Reactor Courses con su código en reactorcourses.com/verificar.

TEMARIO

Las 24 sesiones, abiertas y sin letra chica.

72 horas repartidas en 10 módulos. Así sabes qué verás, qué practicarás y qué entregarás en cada una de las 12 semanas.

MÓDULO 1 **EL CLÚSTER COMO PRODUCTO COMPARTIDO** — 6 h · 2 sesiones

01 **El clúster que usan otros equipos**

el clúster como producto interno: usuarios, contrato y responsabilidades · plano de control desde la operación: API server, scheduler, controller manager y etcd · kubelet, runtime de contenedores y lo que pasa cuando un nodo deja de responder · los tres equipos del programa: Pagos, Campañas y Catálogo · qué decide la plataforma y qué decide cada equipo · versiones de Kubernetes con soporte y su ventana de mantenimiento · kind como laboratorio: qué reproduce de un clúster real y qué no · repositorio de la plataforma y destrucción reproducible del entorno

LABORATORIO: Levantar con kind un clúster con Cilium como CNI y tres nodos de trabajo, detener un nodo y registrar qué componentes reaccionan, en qué orden y cuánto tarda en reprogramarse cada pod

02 **etcd desde la operación: quórum, tamaño y respaldo**

qué guarda etcd y qué no · quórum operativo: cuántos miembros y qué pasa al perder uno o dos · latencia de disco y su efecto en la API · tamaño de la base, compactación y desfragmentación · snapshot de etcd y restauración del plano de control · cifrado de secretos en reposo · plano de control gestionado: qué respalda el proveedor y qué no · señales de un etcd en problemas antes de la caída

LABORATORIO: Tomar un snapshot de etcd, borrar el namespace de un equipo con todos sus recursos y restaurar el clúster al estado previo, midiendo el tiempo hasta que la API vuelve a responder

MÓDULO 2 TENANCY E IDENTIDAD — 9 h · 3 sesiones

03 **Modelos de tenancy y sus límites reales**

namespace por equipo, clúster por equipo y clúster virtual · por qué el namespace no es una frontera de seguridad fuerte · recursos de alcance de clúster: CRD, nodos, clases de almacenamiento · vecino ruidoso: CPU, memoria, red y API server compartidos · clúster virtual: API server propio sobre nodos compartidos · costo y carga operativa de cada modelo · criterios para asignar un modelo a cada equipo · registro de decisión de tenancy con alternativas descartadas

LABORATORIO: Implementar el mismo equipo como namespace con cuotas y como clúster virtual con vCluster, ejecutar seis pruebas de escape —secretos ajenos, CRD globales, nodos, DNS, red y recursos de clúster— y documentar cuáles supera cada modelo

04 **RBAC del clúster y escalamiento de privilegios**

Role, ClusterRole, RoleBinding y ClusterRoleBinding · roles agregados y roles por defecto · verbos peligrosos: escalate, bind e impersonate · permisos que equivalen a administrador: crear pods, leer secretos, nodes/proxy · grupos del proveedor de identidad mapeados a roles · cuentas de servicio y montaje automático del token · revisión de permisos con kubectl auth can-i · RBAC de la plataforma frente a RBAC de cada equipo

LABORATORIO: Diseñar el RBAC de los tres equipos y del operador de plataforma, y demostrar con kubectl auth can-i y diez intentos reales que ningún equipo lee secretos ajenos ni escala privilegios creando pods o usando bind y escalate

05 **Identidad de cargas hacia la nube**

tokens proyectados: audiencia, expiración y vínculo con el pod · el clúster como emisor OIDC y su documento de descubrimiento · Microsoft Entra Workload ID en AKS · EKS Pod Identity e IAM roles for service accounts en EKS · eliminar credenciales de larga vida de los secretos del clúster · una identidad por carga y su alcance mínimo · auditoría del uso de la identidad en el proveedor · presupuesto y alertas en el clúster gestionado del laboratorio

LABORATORIO: Configurar una cuenta de servicio con token proyectado de audiencia y vida acotadas, verificarlo contra el documento OIDC del clúster y demostrar que un pod sin esa cuenta no lo obtiene; extensión opcional en AKS o EKS: leer un secreto de la nube sin ninguna credencial guardada, con presupuesto y alertas configurados y el clúster destruido al cierre

MÓDULO 3 ADMISIÓN, POLÍTICAS Y CADENA DE SUMINISTRO — 9 h · 3 sesiones

06 **Pod Security Standards y Pod Security Admission**

niveles privileged, baseline y restricted · modos enforce, audit y warn por namespace · fijar la versión de la política y por qué · securityContext: usuario, capacidades, sistema de archivos y seccomp · excepciones justificadas y su vencimiento · migrar cargas existentes a restricted sin cortarlas · lo que Pod Security Admission no cubre

LABORATORIO: Aplicar restricted en modo enforce a Pagos y Catálogo y baseline con excepción documentada a Campañas, y demostrar que cinco manifiestos inseguros son rechazados con el mensaje de la política

07 **Políticas de admisión: ValidatingAdmissionPolicy y Kyverno**

ValidatingAdmissionPolicy con CEL: parámetros y vínculos · cuándo alcanza la política nativa y cuándo hace falta un motor · Kyverno: validar, mutar y generar recursos · por qué Kyverno y no OPA Gatekeeper en este programa · generar la NetworkPolicy por defecto al crear un namespace · probar políticas con casos positivos y negativos antes de imponerlas · de auditoría a imposición sin romper cargas existentes · informes de políticas por equipo

`LABORATORIO:` Escribir tres políticas con ValidatingAdmissionPolicy y tres con Kyverno —una de ellas generadora—, probarlas en la integración continua con casos positivos y negativos, y pasarlas de auditoría a imposición sin que falle ninguna carga existente

08 **Cadena de suministro verificada en la admisión**

imagen por digest frente a etiqueta mutable · firma de imágenes con cosign: con clave y sin clave · SBOM con Syft y su adjunto como atestación · verificación de firma en admisión con Kyverno · registro local y política de procedencia por equipo · rotación de la clave de firma · qué protege la firma y qué no protege · excepciones para imágenes de terceros

`LABORATORIO:` Firmar una imagen y su SBOM con cosign, exigir la firma en la admisión con Kyverno y demostrar que se rechazan una imagen sin firmar, una firmada con otra clave y una referenciada por etiqueta mutable

MÓDULO 4 RED DEL CLÚSTER — 9 h · 3 sesiones

09 **CNI y NetworkPolicy entre equipos**

modelo de red de Kubernetes: un IP por pod y su costo · CNI: qué hace y cómo elegirlo · Cilium con eBPF y lo que reemplaza de kube-proxy · denegación por defecto por namespace · excepciones mínimas: DNS, gateway y base de datos · políticas de entrada y de salida · observar flujos permitidos y descartados con Hubble · pruebas de conectividad como requisito de aceptación

`LABORATORIO:` Aplicar denegación por defecto por equipo con excepciones mínimas y demostrar con Hubble y doce pruebas de conectividad que Pagos solo recibe tráfico de su gateway y solo sale hacia su DNS y su base de datos

10 **Gateway API compartido entre equipos**

GatewayClass, Gateway y rutas: quién administra cada recurso · HTTPRoute, GRPCRoute y TLSRoute · ListenerSet para delegar listeners · ReferenceGrant para referencias entre namespaces · terminación TLS y BackendTLSPolicy · retiro de Ingress NGINX y migración con ingress2gateway · Envoy Gateway como implementación · balanceadores en kind con cloud-provider-kind

`LABORATORIO:` Publicar los servicios de los tres equipos con un Gateway compartido donde cada equipo administra solo sus rutas, y demostrar que un equipo no puede adjuntar una ruta al listener de otro ni apuntar a un servicio ajeno sin ReferenceGrant

11 **DNS del clúster y salida controlada**

CoreDNS: resolución, caché y reenvío · ndots y búsquedas en cascada · DNS como punto de falla de todo el clúster · políticas de salida por nombre de dominio · puerta de salida con IP fija para socios externos · servicios externos y endpoints manuales · medir la latencia de resolución por carga

`LABORATORIO:` Restringir la salida de Pagos a dos dominios con una política por nombre de dominio, demostrar que un tercer dominio queda bloqueado y medir con las métricas de CoreDNS el efecto de ndots en la latencia de resolución

MÓDULO 5 ALMACENAMIENTO Y RECUPERACIÓN — 6 h · 2 sesiones

12 **CSI, StatefulSet y snapshots**

CSI: controlador, nodo y clases de almacenamiento · modos de acceso y política de recuperación del volumen · StatefulSet: identidad estable y orden de arranque · topología del volumen y planificación del pod · VolumeSnapshot y VolumeSnapshotClass · expandir un volumen en uso · cuándo una base de datos no debería correr en el clúster

`LABORATORIO:` Desplegar una base de datos con StatefulSet sobre el driver CSI hostpath, tomar un VolumeSnapshot, corromper los datos y restaurar desde el snapshot verificando el conteo de registros

13 **Respaldo y restauración con Velero**

qué respalda Velero: recursos, volúmenes y metadatos · almacén de objetos compatible con S3 en local con SeaweedFS · por qué no MinIO: repositorio archivado y licencia · respaldo programado, retención y alcance por equipo · restauración en otro clúster y mapeo de namespaces · objetivos de tiempo y de punto de recuperación por equipo · lo que Velero no cubre: etcd, CRD externos y secretos de la nube · la prueba de restauración como requisito

LABORATORIO: Respaldar con Velero el namespace de Pagos hacia un almacén S3 local y restaurarlo en un clúster nuevo, midiendo el tiempo de recuperación y verificando datos, políticas y permisos

MÓDULO 6 CAPACIDAD Y PLANIFICACIÓN — 9 h · 3 sesiones

14 **Requests, límites, QoS y expulsiones**

requests y limits: qué usa el scheduler y qué usa el kernel · clases QoS: Guaranteed, Burstable y BestEffort · expulsión por presión de memoria y de disco · OOMKilled frente a expulsión · ResourceQuota y LimitRange por equipo · PriorityClass y desalojo preventivo · límite de CPU y su efecto en la latencia · explicar un incidente de capacidad con eventos

LABORATORIO: Provocar una expulsión por presión de memoria en un nodo y explicarla con los eventos y la clase QoS de cada pod; demostrar que la PriorityClass de Campañas protege su carga crítica

15 **Planificación: taints, afinidad y disrupciones**

taints y tolerations para nodos dedicados · afinidad y antiafinidad de nodo y de pod · topology spread constraints entre zonas · PodDisruptionBudget y desalojo voluntario · drain, cordon y mantenimiento de nodos · grupos de nodos por equipo frente a nodos compartidos · fragmentación de recursos en el clúster

LABORATORIO: Dedicar un grupo de nodos a Pagos con taints y afinidad, repartir Campañas entre zonas simuladas con topology spread y demostrar que un drain respeta los PodDisruptionBudget de cada equipo

16 **Autoescalado de pods y de nodos**

HPA: métricas, objetivo y comportamiento de escalado · VPA en modo recomendación y sus riesgos en modo automático · Cluster Autoscaler: grupos de nodos y escalado a cero · Karpenter y aprovisionamiento por pod pendiente · Cluster API como forma de describir y escalar nodos · tiempo de reacción de cada capa ante un pico · sobreaprovisionamiento con pods de baja prioridad

LABORATORIO: Montar un clúster con Cluster API sobre Docker, aplicar una carga a Campañas y demostrar que HPA escala los pods y Cluster Autoscaler agrega un nodo y lo retira al bajar la carga, con el tiempo de cada etapa medido

MÓDULO 7 GITOPS Y AUDITORÍA DE CAMBIOS — 6 h · 2 sesiones

17 **Argo CD para varios equipos**

el clúster reconciliado desde Git · por qué Argo CD y no Flux en este programa · AppProject por equipo: destinos, orígenes y recursos permitidos · ApplicationSet para generar aplicaciones por equipo · sincronización, poda y autocorrección del desvío · orden de despliegue con sync waves · acceso de los equipos a Argo CD y su RBAC propio

LABORATORIO: Entregar la configuración de los tres equipos con Argo CD y un AppProject por equipo, y demostrar que un cambio manual en el clúster se revierte solo y que un equipo no puede desplegar fuera de sus namespaces

18 **Promoción entre entornos y registro de auditoría**

estructura del repositorio: base y entornos con Kustomize · promoción por pull request y revisión obligatoria · política de auditoría de la API: niveles y etapas · quién cambió qué: Git, Argo CD y registro de auditoría · cambios de emergencia fuera de Git y su reconciliación · retención y costo de los registros de auditoría · secretos en GitOps sin guardarlos en claro

LABORATORIO: Promover un cambio de desarrollo a producción por pull request y reconstruir, con el registro de auditoría de la API y el historial de Git, quién cambió qué y cuándo en un incidente simulado

MÓDULO 8 OBSERVABILIDAD Y COSTO POR EQUIPO

6 h · 2 sesiones

19 Observabilidad del clúster y línea base de seguridad

métricas del API server: latencia, errores y solicitudes en vuelo · métricas de etcd y de los nodos · kube-state-metrics y eventos del clúster · Prometheus y alertas sobre el plano de control · registros de auditoría como señal de seguridad · CIS Kubernetes Benchmark con kube-bench · qué hallazgos del benchmark no aplican a kind ni a un clúster gestionado

LABORATORIO: Instrumentar el clúster con Prometheus, crear cinco alertas del plano de control y de los nodos, verificar que disparan ante una falla inducida y corregir dos hallazgos de kube-bench que sí aplican

20 Costo del clúster por equipo

OpenCost: asignación por namespace y por etiqueta · costo compartido de la plataforma y su reparto · sobreasignación: requests frente a uso real · precios de referencia de nodos en la nube · recomendaciones del VPA como evidencia · informe de costo para cada equipo · optimizar sin romper la disponibilidad

LABORATORIO: Atribuir con OpenCost el costo del clúster a los tres equipos con precios de referencia, identificar la sobreasignación del equipo con más holgura y proponer requests corregidos respaldados por la recomendación del VPA

MÓDULO 9 CICLO DE VIDA Y ACTUALIZACIONES

6 h · 2 sesiones

21 Calendario de versiones y APIs que desaparecen

calendario de lanzamientos y fin de soporte de cada versión menor · política de diferencia de versiones entre componentes · APIs obsoletas y APIs eliminadas · detectar APIs eliminadas en el clúster y en Git · versiones de los complementos: CNI, Gateway, Kyverno y Argo CD · soporte extendido en clústeres gestionados y su costo · plan de actualización: ventana, criterios de aborto y reversión

LABORATORIO: Auditar el clúster y el repositorio en busca de APIs que se eliminan en la versión siguiente, corregir los manifiestos en Git y escribir el plan de actualización con ventana, criterios de aborto y reversión

22 Actualización en vivo con Cluster API

orden de actualización: plano de control, complementos y nodos · reemplazo de nodos con surge frente a actualización en sitio · drain y PodDisruptionBudget durante la actualización · Cluster API: KubeadmControlPlane y MachineDeployment · verificación posterior: conformidad mínima y pruebas de humo · qué se puede revertir y qué no después de actualizar etcd · comunicar la ventana a los equipos

LABORATORIO: Actualizar una versión menor un clúster de Cluster API sobre Docker con carga en curso, sin violar ningún PodDisruptionBudget, y medir las peticiones fallidas durante toda la actualización

MÓDULO 10 PROYECTO INTEGRADOR

6 h · 2 sesiones

23 Game day de la plataforma

diseño del ejercicio: hipótesis, alcance y criterio de corte · falla 1: nodo perdido con carga de Pagos · falla 2: cuota agotada en Campañas durante un pico · detección por alertas y no por reclamo · manual de respuesta por síntoma · tiempo de detección y tiempo de recuperación · revisión posterior sin buscar culpables

LABORATORIO: Ejecutar un game day sobre la plataforma de los tres equipos con dos fallas inyectadas —nodo perdido y cuota agotada—, detectarlas por alertas y resolverlas con el manual, registrando tiempos de detección y de recuperación

24 Defensa de la plataforma

decisiones de tenancy, políticas y red con sus alternativas · actualización de versión menor en vivo · plan de reversión y criterio de aborto · informe de costo por equipo · resultado del game day y correcciones aplicadas · destrucción y reconstrucción del entorno desde Git · preguntas cruzadas del grupo

LABORATORIO: Defender la plataforma ante el grupo con una actualización de versión menor en vivo y su plan de reversión, el informe de costo por equipo y el resultado del game day

LO QUE VAS A USAR

Herramientas elegidas porque resuelven una decisión de la plataforma, no por estar de moda.

Laboratorio

- kind
- Docker
- cloud-provider-kind
- Cluster API con el proveedor Docker

Tenancy e identidad

- RBAC y cuentas de servicio
- vCluster
- Tokens proyectados y emisor OIDC del clúster
- Workload ID de AKS o Pod Identity de EKS (opcional)

Admisión

- Pod Security Admission
- ValidatingAdmissionPolicy
- Kyverno

Cadena de suministro

- cosign
- Syft
- Registro de imágenes local

Red

- Cilium y Hubble
- Gateway API con Envoy Gateway
- CoreDNS

Almacenamiento y recuperación

- Driver CSI hostpath y VolumeSnapshot
- Velero
- SeaweedFS como almacén S3 local
- etcdctl

Capacidad

- ResourceQuota, LimitRange y PriorityClass
- HPA y metrics-server
- VPA
- Cluster Autoscaler

GitOps

- Argo CD
- Kustomize
- Registro de auditoría de la API

Observabilidad y costo

- Prometheus y kube-state-metrics
- kube-bench
- OpenCost

Los logos son marcas de sus respectivos dueños. Se muestran solo para identificar las herramientas que se usan en el programa y no implican aval, patrocinio ni afiliación.

CÓMO FUNCIONA

Clases en vivo, un clúster en tu máquina y cada control probado intentando romperlo.

01 **Clases en vivo por Zoom** 24 sesiones de 3 horas, 2 veces por semana. Se trabaja sobre el clúster de cada uno y se revisan las decisiones con el grupo.

02 **Todo queda grabado** Si te pierdes una sesión o vives en otro huso horario, la grabación queda disponible. Acceso por 12 meses.

03 **30% concepto / 70% laboratorio** Se explica el mecanismo y su límite, y enseguida se configura sobre la plataforma de los tres equipos.

04 **Prueba negativa obligatoria** Ningún control de seguridad se da por bueno porque se aplicó: se aprueba cuando el intento que debía impedir fue intentado y rechazado.

05 **Todo corre en tu máquina** kind y Cluster API sobre Docker, con software de código abierto. Solo una extensión opcional usa un clúster gestionado, con presupuesto y alertas, y se destruye al terminar.

06 **Termina con fallas reales** Un game day con dos fallas inyectadas y una actualización de versión en vivo, defendidas ante el grupo.

Horarios · Primer grupo

Miércoles	19:00 - 22:00
Viernes	19:00 - 22:00

Horario referido a GMT-5 (Perú, Colombia, Ecuador y Panamá; en Ciudad de México es una hora menos).

Los días y el horario son la propuesta para este grupo y se confirman junto con la fecha de inicio, antes de abrir la matrícula. El calendario definitivo se entrega al confirmar la matrícula.

REQUISITOS

Lo que necesitas traer a la sesión 01.

No hay nivelación previa. La sesión 01 levanta un clúster con varios nodos en tu máquina y empieza a romperlo en la primera hora.

Kubernetes

- ✓ Haber desplegado al menos una aplicación en un clúster
- ✓ Deployment, Service, Ingress, ConfigMap y Secret
- ✓ kubectl con soltura
- ✓ Leer y escribir manifiestos YAML

Redes

- ✓ TCP/IP, subredes y puertos
- ✓ DNS y resolución de nombres
- ✓ TLS a nivel práctico
- ✓ Diagnóstico básico de conectividad

Contenedores y Linux

- ✓ Imagen, registro y capas
- ✓ Docker en tu máquina
- ✓ Terminal de Linux
- ✓ Procesos, permisos y sistema de archivos

Trabajo diario

- ✓ Git y pull requests
- ✓ Nociones de integración continua
- ✓ Nociones de una nube pública

Equipo y tiempo

- ✓ Computadora con 16 GB de memoria y 4 núcleos
- ✓ Seis horas de clase por semana
- ✓ Entre cuatro y seis horas de laboratorio
- ✓ 12 semanas de compromiso

Costos de herramientas que asumes tú `US$0 a 25 aprox.`

Estos costos NO están incluidos en el precio del programa. Todo el software del laboratorio es de código abierto y gratuito.

- El laboratorio corre en tu máquina con kind y Docker: con un equipo de 16 GB de memoria el costo de nube es cero
- Si tu equipo no alcanza, una máquina virtual de 4 núcleos y 16 GB alquilada solo durante las prácticas cuesta del orden de US$20 en todo el programa
- Una única extensión opcional en un clúster gestionado (AKS o EKS) en la sesión 05: unos pocos dólares, con presupuesto y alertas configurados en esa sesión y el clúster destruido al cierre

PRECIO

72 horas para administrar el clúster que usan otros, y probar que aguanta.

US$400

Pago único · USD · Incluye impuestos

Todavía no hay fecha de inicio. Escríbenos y te avisamos cuando la confirmemos: no cobramos nada antes de abrir la matrícula.

Incluye

- ✓ 24 sesiones en vivo por Zoom (72 horas lectivas)
- ✓ Grabación de todas las clases con acceso por 12 meses
- ✓ Repositorio de la plataforma con el punto de partida de cada laboratorio
- ✓ Manifiestos de los tres equipos del caso y sus cargas de prueba
- ✓ Banco de pruebas negativas de seguridad y de red
- ✓ Guion del game day y de la actualización en vivo
- ✓ Los 3 certificados progresivos
- ✓ Canal privado del grupo para dudas entre sesiones
- ✓ Actualizaciones del material por 12 meses

No incluye

- – El equipo de trabajo: necesitas una computadora con 16 GB de memoria; si no la tienes, alquilar una máquina virtual entra en el rango de costos propios (US$0 a 25)
- – Exámenes ni certificaciones de terceros: este programa no prepara para ninguno
- – Licencias comerciales: todo el laboratorio corre con software de código abierto

Métodos de pago

Tarjeta de crédito · Tarjeta de débito internacional · PayPal · Apple Pay · Google Pay

Garantía de 14 días. Si dentro de los primeros 14 días del programa decides que no es para ti, te devolvemos el 100 % de lo pagado. No pedimos justificación.

Paddle actúa como comerciante registrado (merchant of record): emite el comprobante y gestiona los impuestos de cada país.

PREGUNTAS FRECUENTES

Preguntas frecuentes sobre el programa.

¿Las clases son en vivo o grabadas?

En vivo por Zoom: 24 sesiones de 3 horas, 2 veces por semana. Todas quedan grabadas y disponibles por 12 meses.

¿Cuándo empiezan las clases?

La fecha de inicio y el calendario definitivo todavía no están confirmados: se publican en esta misma ficha antes de que se abra la matrícula. Los días propuestos son Miércoles y viernes en horario de GMT-5 (Perú, Colombia, Ecuador y Panamá; en Ciudad de México es una hora menos). Si quieres que te avisemos cuando se confirmen, escribe a soporte@reactorcourses.com.

¿Qué nivel necesito?

Haber desplegado aplicaciones en Kubernetes y manejar kubectl y YAML sin dificultad. El programa no explica qué es un Deployment: empieza administrando el clúster. Si nunca trabajaste con Kubernetes, este programa te va a quedar grande.

¿Prepara para CKA, CKAD o CKS?

No. Esos exámenes tienen su propio temario y formato, y este programa no los sigue. Muchos temas se cruzan —RBAC, red, almacenamiento, seguridad—, pero aquí se tratan como decisiones de una plataforma compartida, no como tareas cronometradas de examen.

¿Necesito una cuenta de nube?

No. Todo el laboratorio corre en tu máquina con kind y Cluster API sobre Docker. Hay una única extensión opcional en un clúster gestionado de Azure o AWS para la identidad de cargas, que cuesta unos pocos dólares si la haces y se destruye al terminar la sesión.

¿Qué equipo necesito?

Una computadora con 16 GB de memoria y 4 núcleos, con Docker instalado. Algunos laboratorios levantan varios clústeres a la vez. Si tu equipo no alcanza, puedes alquilar una máquina virtual solo durante las prácticas. El rango de costos propios del programa es de US$0 a 25: cero si tu equipo alcanza y del orden de 20 dólares si alquilas la máquina.

Ya cursé un programa de plataforma de nube. ¿Me sirve este?

Sí, pero conviene que sepas qué se repite. Cerca de un tercio de las sesiones trata problemas que un programa de plataforma de nube aborda a nivel de cuenta: menor privilegio, identidad de cargas, segmentación de red, salida controlada, políticas preventivas, auditoría de cambios, observabilidad y costo por equipo. Aquí se resuelven dentro del clúster, con RBAC, admisión, NetworkPolicy, Argo CD y OpenCost, y con otros laboratorios. Lo nuevo serían tenancy, etcd, capacidad, almacenamiento, Gateway API y actualizaciones. No hay descuento ni convalidación entre programas.

¿Por qué Kyverno y no OPA Gatekeeper, o Argo CD y no Flux?

Porque hay que elegir uno para profundizar, y en los dos casos elegimos por la decisión que resuelve en una plataforma compartida: Kyverno valida, muta, genera y verifica firmas con políticas en YAML; Argo CD tiene proyectos por equipo que encajan con el modelo de tenancy. Se explica en qué casos la otra opción sería mejor.

¿Qué versión de Kubernetes se usa?

Una versión con soporte del proyecto al momento de dictar el programa, y la actualización en vivo se hace hacia la siguiente versión menor también con soporte. Las versiones exactas se confirman al publicar el calendario, porque cada versión menor tiene alrededor de un año de soporte.

¿Se enseña service mesh?

Solo como decisión: cuándo justifica su costo y qué resuelve que la red del clúster no. No se instala ni se opera aquí.

¿Qué pasa si falto a una clase?

La grabación queda disponible y cada laboratorio tiene su punto de partida en el repositorio, así que puedes ponerte al día antes de la siguiente sesión. La plataforma se construye por capas: faltar a varias seguidas sin recuperarlas te deja sin las piezas que usa el proyecto final.

¿Los certificados tienen validez oficial?

No. Son certificados de Reactor Courses que acreditan las horas y competencias del programa. No son títulos universitarios ni certificaciones de la Linux Foundation o de la CNCF, y no tenemos ningún aval de esas organizaciones.

¿Cuánto tiempo debo dedicarle por semana?

Seis horas de clase más entre cuatro y seis horas de laboratorio. Considera entre 10 y 12 horas semanales durante 12 semanas.

¿Qué métodos de pago aceptan?

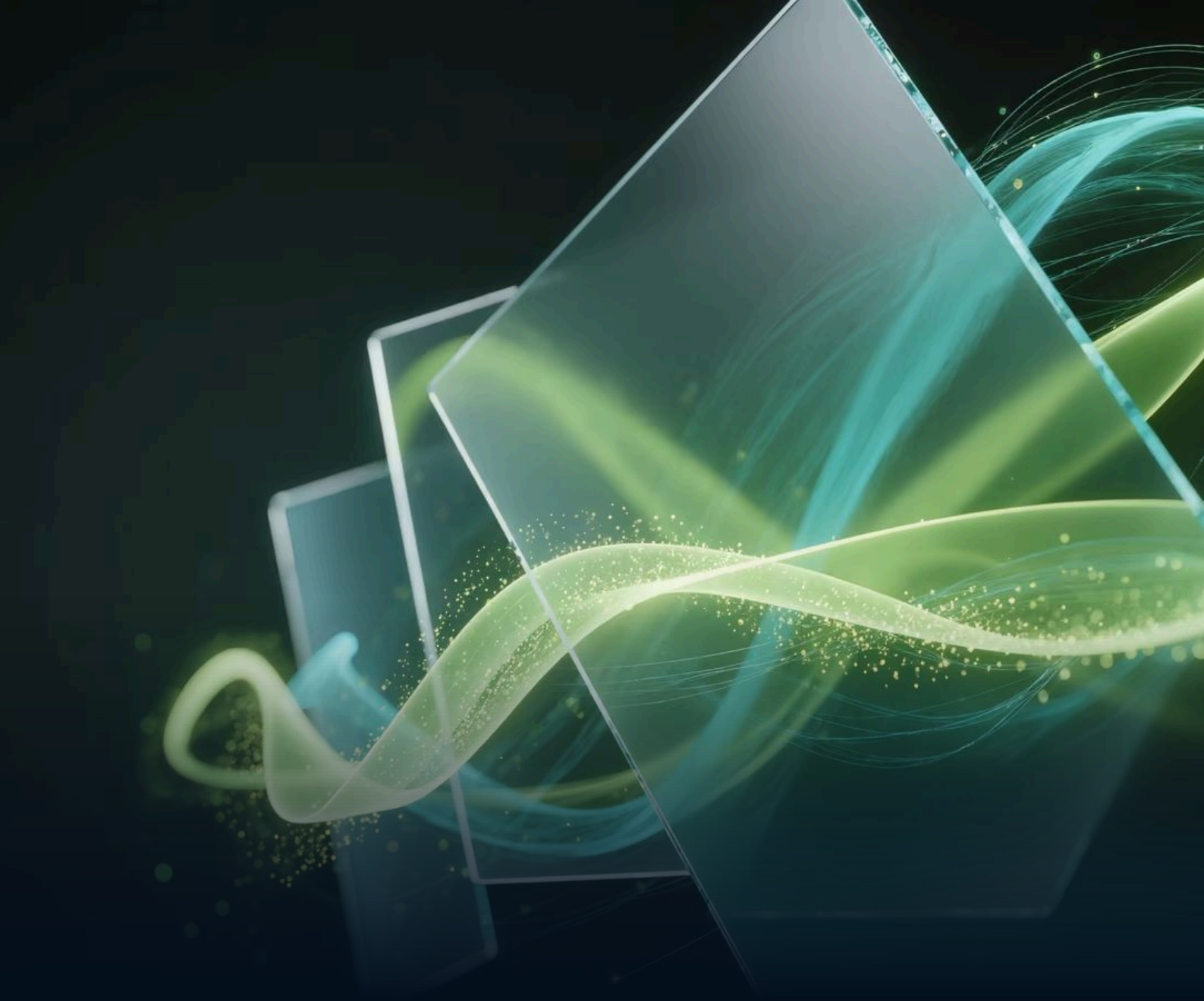
El pago lo procesa Paddle y acepta Tarjeta de crédito, Tarjeta de débito internacional, PayPal, Apple Pay, Google Pay. El cobro es en dólares. Importante: no procesamos Yape, Plin, OXXO, Mercado Pago ni transferencias locales. Si tu banco rechaza compras internacionales, habilítalas antes de pagar o usa PayPal.

¿Emiten factura o comprobante?

Sí. Paddle actúa como comerciante registrado (merchant of record): emite el comprobante y gestiona los impuestos de cada país. Lo recibirás por correo automáticamente al completar el pago.

¿Cuál es la política de reembolso?

Tienes 14 días desde el inicio del programa para pedir el reembolso completo escribiendo a soporte@reactorcourses.com. Publicamos el temario completo y los requisitos justamente para que decidas informado.



Reactor
COURSES

Ingeniería de software en la era de los agentes

¿Te quedó alguna duda?

Escríbenos a info@reactorcourses.com. La ficha al día, con el temario interactivo y la inscripción, está en reactorcourses.com/programas/kubernetes-platform-architecture.

`reactorcourses.com/programas/kubernetes-platform-architecture`

Información vigente al 23 de noviembre de 2026. El precio, las fechas y el temario pueden cambiar: la versión al día es la de la web.

Reactor Courses no está afiliado a Microsoft, Amazon Web Services, OpenAI, Anthropic ni Google. Los nombres de productos se mencionan únicamente con fines descriptivos del contenido académico.