

ESNOG 2026 · BARCELONA

Una nube soberana para operadores

Red, cómputo e IA sobre Kubernetes, con todo en git

José Manuel Román

Fibercli · Encinia

CLI · Backstage · agentes MCP

una API: Kubernetes + XRD

VM · S3 · Postgres · LLM · agentes

Capsule · Crossplane · Kueue

Ceph · KubeVirt · vLLM

RKE2 · Cilium · Multus

hierro propio

Tres partes, sin relleno

01

Arquitectura

Capas, red, seguridad, multi-tenant, IA y GitOps. Y diez buenas prácticas, explicadas.

slides 4-39

02

API, CLI y dashboard

Un modelo de recursos, una API granular delante (api-organizations) y cuatro clientes: encinia-cli, Backstage, agentes y clústeres distribuidos.

slides 40-55

03

Tres casos de uso

Laboratorio de operador con CHR, entornos GPU para docentes e investigadores, plataforma agéntica.

slides 56-76

Una API. Cuatro clientes.

 encinia-cli

 Backstage

 agentes MCP

 clústeres distribuidos

Cada servicio es un recurso declarativo de Kubernetes. Lo que pide un humano por la CLI, lo que propone un agente y lo que sincroniza un clúster distribuido como rke06 salen del mismo modelo y pasan por los mismos controles.

PARTE 1

01

Arquitectura

Del hierro a la API: red, seguridad, tenants, servicios, IA y GitOps.

Por qué esto interesa a un operador



Casos de negocio reales

Hay demanda real de cómputo, GPU e IA de clientes que no pueden sacar sus datos fuera.



IA en la operación

Diagnóstico, auditoría y documentación ya se hacen con modelos. Mejor si no salen del CPD.



Ya tenemos red y CPD

Fibra, tránsito, energía y personal de nivel 3. Falta la capa de plataforma.



ENS, NIS2 y AI Act

Clientes públicos y regulados piden datos en territorio nacional, evidencias de control y trazabilidad de la IA.

Cuatro reglas que no negociamos



R1

Todo declarativo en git

Lo que se toca a mano se lleva a git o caduca. Si no está en git, no existe.



R2

Red por API

Routers y firewalls por API. La CLI solo para el bootstrap mínimo.



R3

Soberanía

Open source, on-premise y con air-gap posible.

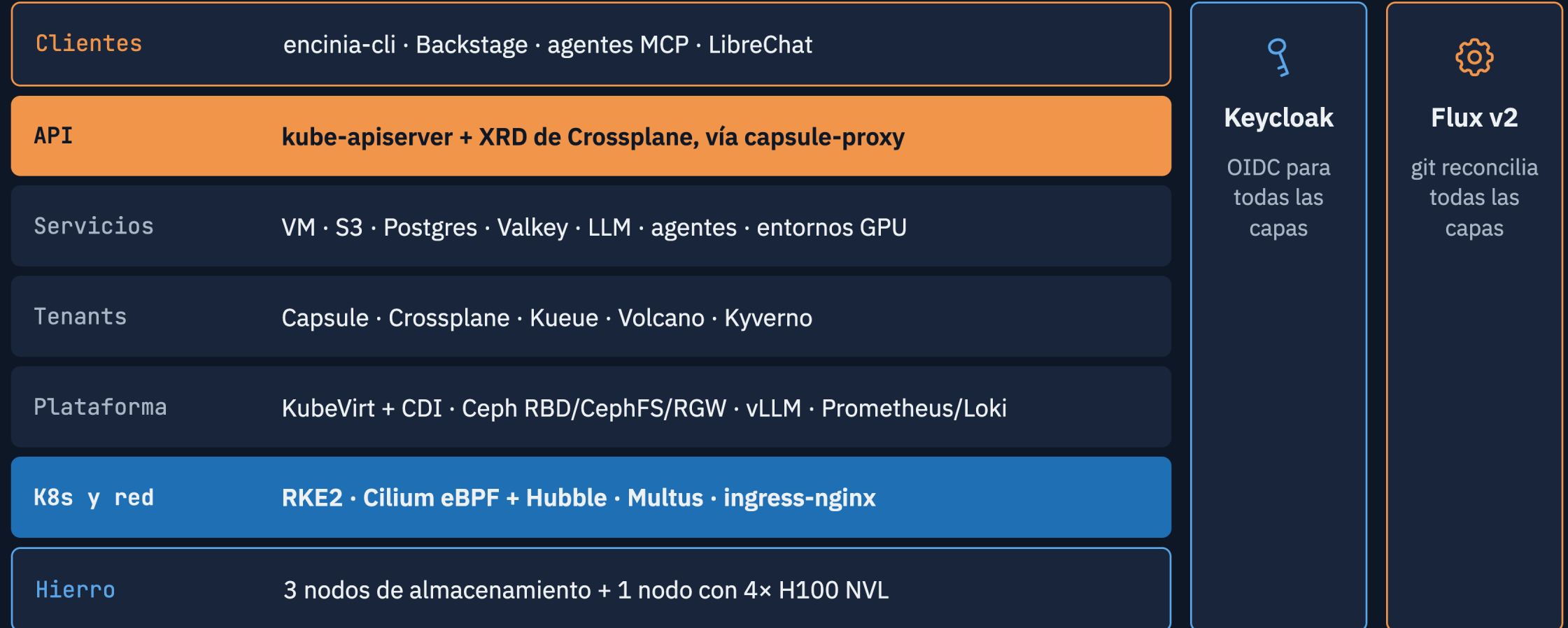


R4

Cumplimiento desde el diseño

ENS (RD 311/2022), NIS2 y AI Act como requisito, no como auditoría a posteriori.

La pila completa, de abajo arriba



Qué ofrecemos, por categoría de servicio

Capa	Software libre que usamos	Categoría de servicio equivalente
Red	Cilium + Hubble, Multus, ingress-nginx	redes virtuales, interfaces secundarias, registro de flujos
Seguridad	Tetragon, Kyverno, cosign	detección en ejecución, políticas, firma de imágenes
Identidad	Keycloak + provider-keycloak	directorío e inicio de sesión único
Tenants	Capsule, Crossplane, Kueue	cuentas por cliente y cuotas de servicio
Cómputo	KubeVirt + CDI	máquinas virtuales bajo demanda
Almacén	Ceph RBD, CephFS, RGW	bloque, ficheros compartidos y objetos
Backup	Velero + Kopia, barman-cloud	copias de seguridad gestionadas
Datos	CNPG, MongoDB, ClickHouse, Valkey	bases de datos y caché gestionadas

IA, observabilidad y entrega

Capa	Software libre que usamos	Categoría de servicio equivalente
Inferencia	vLLM + LiteLLM	API de modelos de lenguaje
Agentes	LangGraph, HolmesGPT, MCP	ejecución de agentes
Entornos	EntornoEntrenamiento (XRD)	entornos de entrenamiento gestionados
Observ.	Prometheus, Thanos, Loki, Grafana	métricas, logs y trazas
Coste	OpenCost, DCGM	análisis de costes
Supply chain	Harbor + Trivy, Forgejo Runner	registro de imágenes, escaneo e integración continua
Portal	Backstage, Headlamp, LibreChat	consola y catálogo de servicios
GitOps	Flux v2, Rancher + Fleet + CAPI	infraestructura como código, Kubernetes gestionado

Lo que corre hoy, todo reconciliado desde git

13

Kustomizations de Flux, todas Ready

29

HelmReleases gestionadas

10

tenants de Capsule

6

XRD propias de Crossplane

7

Postgres con backup diario y
WAL continuo

75

objetos de Keycloak declarados en git

eBPF para la red de pods, Hubble como NetFlow

tenant-a/web

tenant-a/db

tenant-b/api

Datapath eBPF de Cilium

Identidad por etiquetas, NetworkPolicy L3–L7 y balanceo de servicios en el kernel



Hubble exporta cada flujo con su identidad

NIC del nodo · ingress-nginx · CoreDNS

```
$ hubble observe -n tenant-a
```

```
tenant-a/web → tenant-a/db  
TCP 5432 · FORWARDED
```

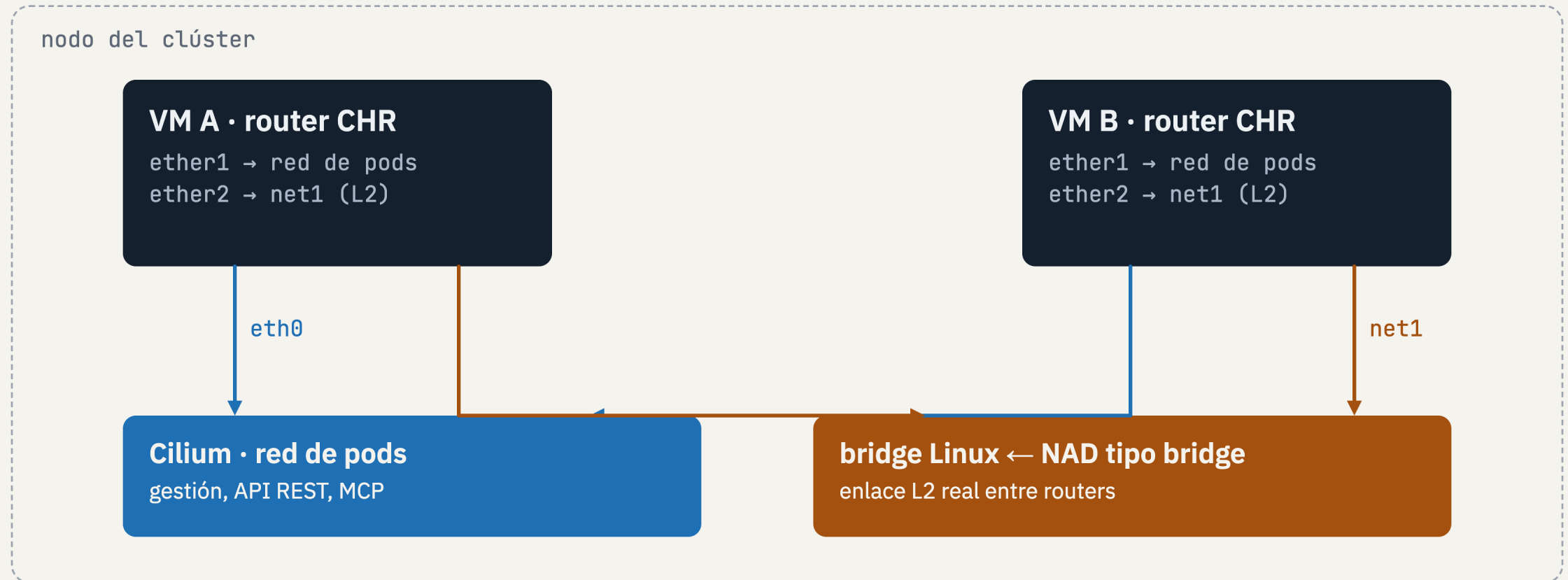
```
tenant-a/web → kube-dns  
UDP 53 · FORWARDED
```

```
tenant-b/api → tenant-a/db  
TCP 5432 · DROPPED (policy denied)
```

```
tenant-a/web → world  
TCP 443 · DROPPED (egress)
```

Salida ilustrativa, nombres genéricos

Una segunda interfaz L2 para VMs y pods



Dos ficheros y un bridge

1 · Nodo: Multus delante de Cilium

```
# config.yaml.d/50-multus.yaml
cni:
  - multus
  - cilium

# HelmChartConfig rke2-cilium
valuesContent: |-
  cni:
    exclusive: false
```

2 · Plataforma: la NAD, en el namespace del tenant

```
kind: NetworkAttachmentDefinition
metadata:
  name: br-lab-l1
  namespace: operador001-lab
spec:
  config: |
    {"cniVersion": "0.3.1",
     "type": "bridge",
     "bridge": "br-lab-l1",
     "ipam": {}}
```

Sin **exclusive: false**, Cilium reescribe la configuración CNI y Multus desaparece del nodo.

Dos capas: qué entra y qué hace una vez dentro



Admisión · Kyverno

antes de crear el objeto

- Firma cosign de la imagen
- Procedencia SLSA nivel 1
- Políticas de red y de tenant
- Rechaza: el pod nunca existe



pod

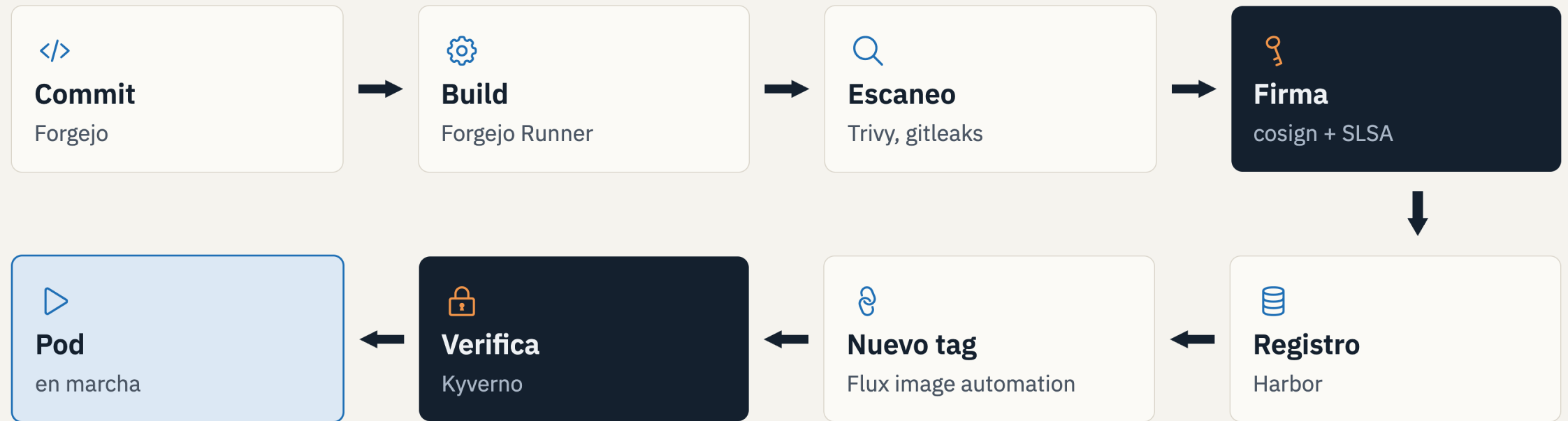


Runtime · Tetragon

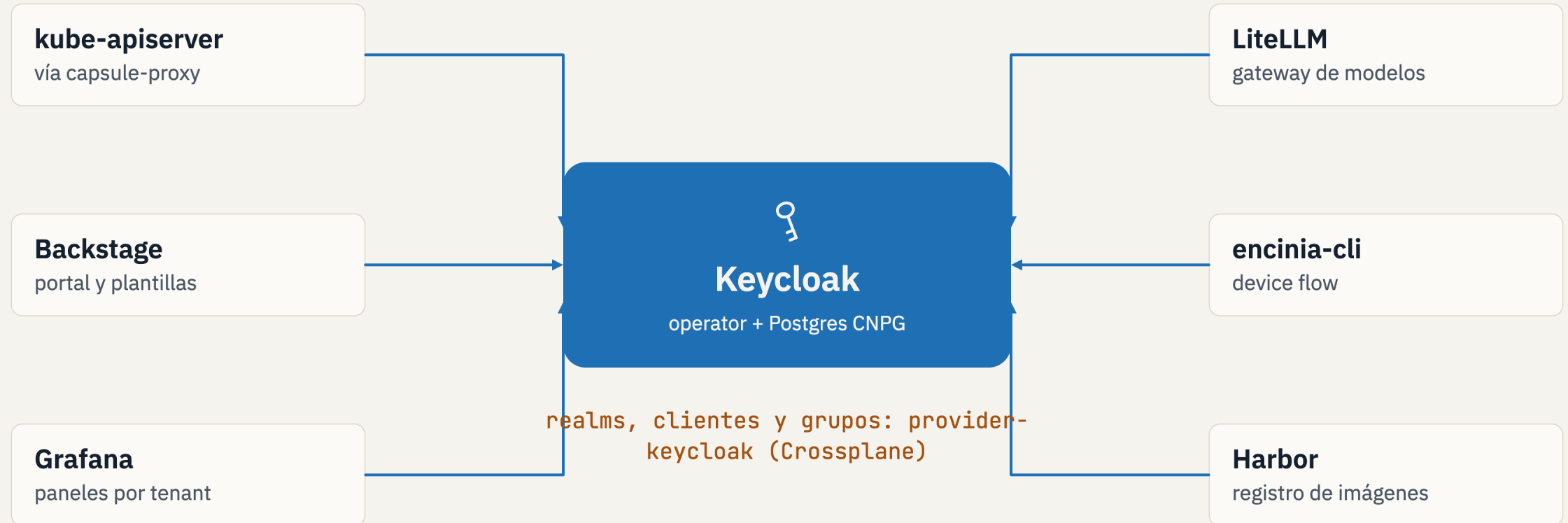
mientras se ejecuta

- Procesos y syscalls con eBPF
- Ficheros sensibles y red
- Eventos a Loki con identidad del pod
- Puede matar el proceso en el kernel

Del commit al pod: firmado en CI, verificado en admisión



Un único issuer OIDC para todo



Capsule agrupa; capsule-proxy filtra

Tenant operador001



heredado en todos los namespaces

ResourceQuota

LimitRange

NetworkPolicy

RBAC

directo al apiserver

```
$ kubectl get namespaces
```

Error: forbidden (cluster scope)

a través de capsule-proxy

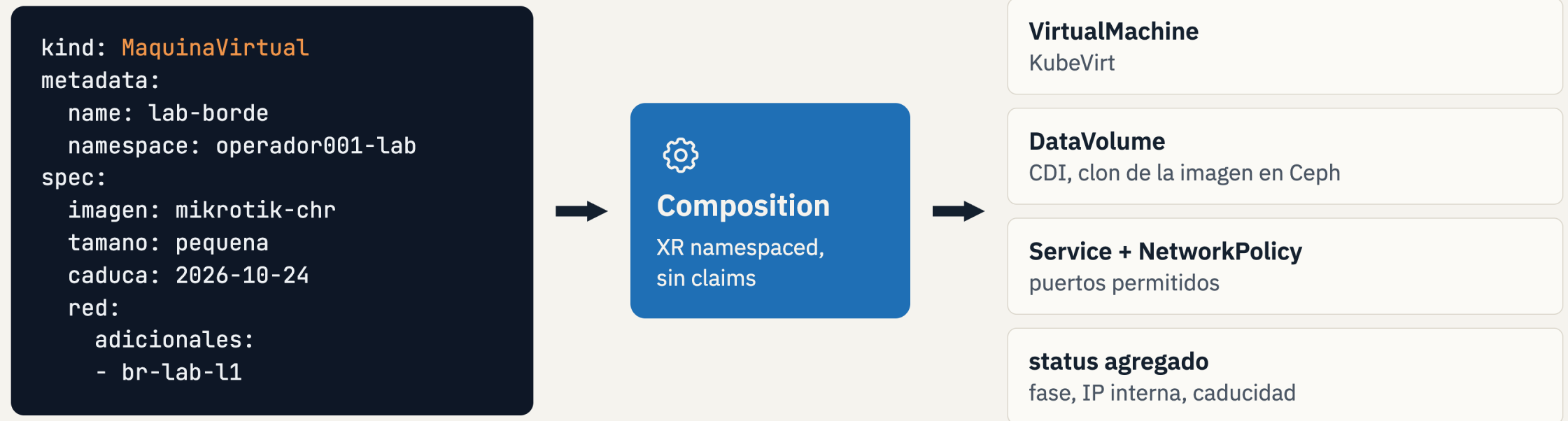
```
$ kubectl get namespaces
```

```
operador001-red
```

```
operador001-agentes
```

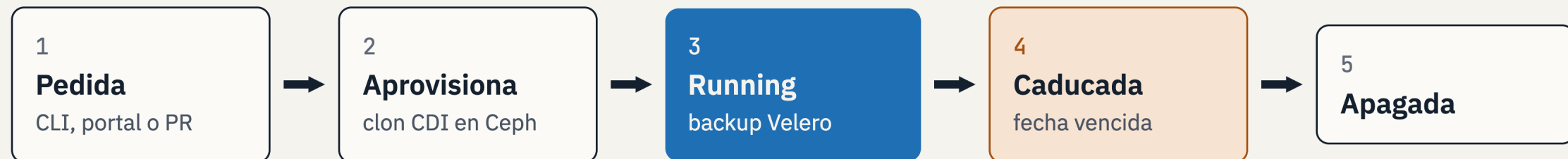
```
operador001-lab ...
```

El usuario pide quince líneas; la plataforma compone el resto



XRD propias: `MaquinaVirtual` · `AgentRuntime` · `EntornoEntrenamiento` · `CuotaGPU`

Ciclo de vida de una MaquinaVirtual



Caducidad en el recurso

La fecha de caducidad y la política de backup van en la propia MaquinaVirtual, en git.

Catálogo

debian-13 · rocky-9 · ubuntu-24.04 · fedora ·
mikrotik-chr-7.24.5

Ceph para bloque, fichero y objeto

RBD

discos de VM
y PVC

CephFS

volúmenes
compartidos

RGW

objetos de la
plataforma

clúster Ceph · 3 nodos

OSD · MON

OSD · MON

OSD · MON

PVC en Ceph por clase de almacenamiento

ceph-block

25

ceph-fs

4

En RGW van las copias de Velero, el WAL de Postgres con barman-cloud, los bloques de Thanos y las trazas de Langfuse.

Un día de copias, en una línea

Postgres · WAL

archivado continuo a RGW, sin huecos

Postgres · base



01:00–01:30 UTC · 7 clústeres escalonados

VMs · Velero

Velero + Kopia · VMs seleccionadas por etiqueta

00

06

12

18

24

CNPG + barman-cloud

Plugin como sidecar; bucket y usuario RGW por clúster.

7 de 7 archivando

ContinuousArchivingSuccess y primera copia completada.

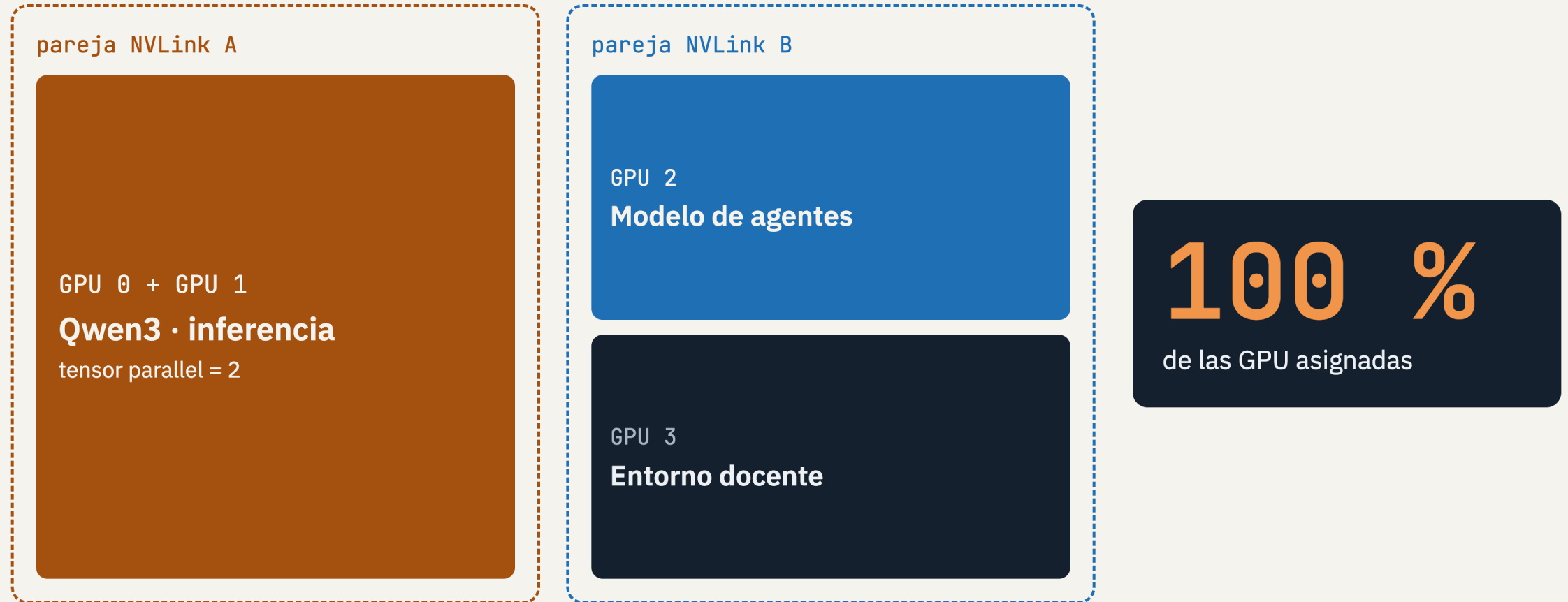
Pendiente

Prueba de restauración y copia fuera del sitio.

Nadie habla con vLLM directamente

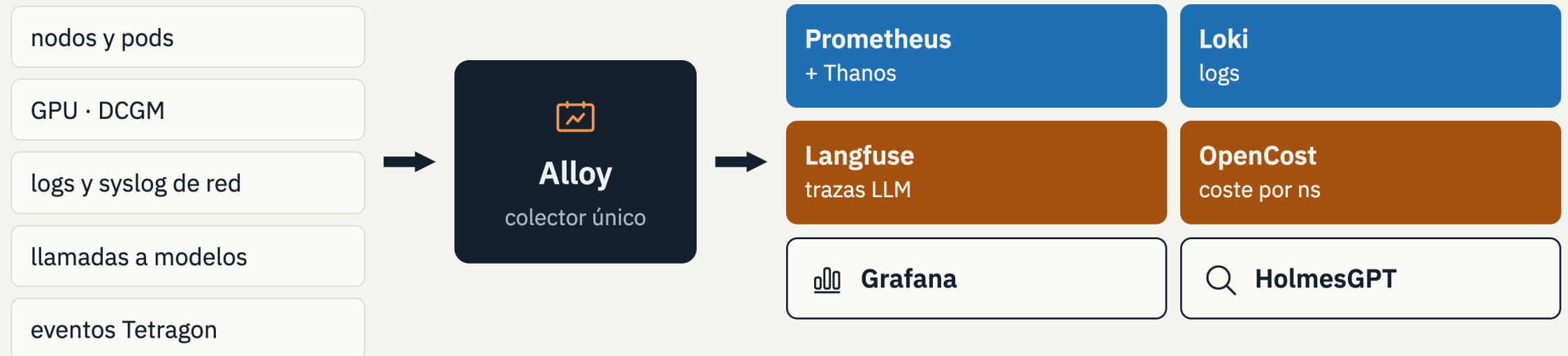


Cuatro GPU, cuatro asignadas



Métricas, logs, trazas de LLM y coste

fuentes



Flux por capas, con dependencias explícitas



Buenas prácticas

Diez prácticas de operador aplicadas a una plataforma de IA: qué hacemos, por qué y cómo.

Cambios

Acceso

Datos y ciclo de vida

Operación de nodos

Diez prácticas en cuatro familias

Cambios	Acceso	Datos y ciclo de vida	Operación de nodos
01 GitOps declarativo	04 Mínimo privilegio	07 Copia continua y réplicas	10 Plano de control impar
02 Cambios escalonados	05 Supply chain fijada	08 Todo caduca	
03 Guardas de contexto	06 Secretos fuera de git	09 Cada cambio deja evidencia	

Todo el estado en git; Flux lo reconcilia

POR QUÉ

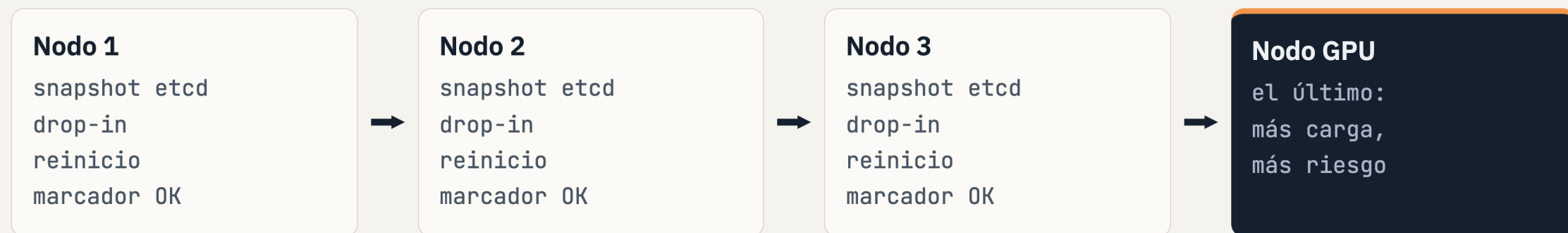
Cada cambio se revisa, se puede revertir con un git revert y deja evidencia de quién lo hizo. El clúster se puede reconstruir desde cero.

CÓMO

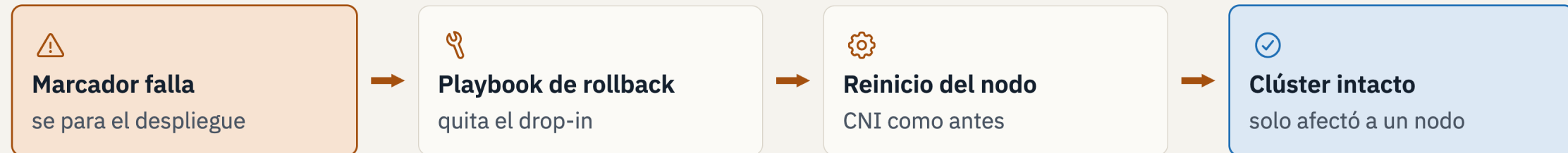
- Capas con dependencias explícitas
- Sin borrado automático en infraestructura
- Raíz con lista explícita: nada entra sin declararlo
- Recursos con dependencias raras, en su propia capa



Cambios escalonados, con la vuelta atrás escrita antes



SI UN NODO NO PASA SU MARCADOR · VUELTA ATRÁS PROBADA



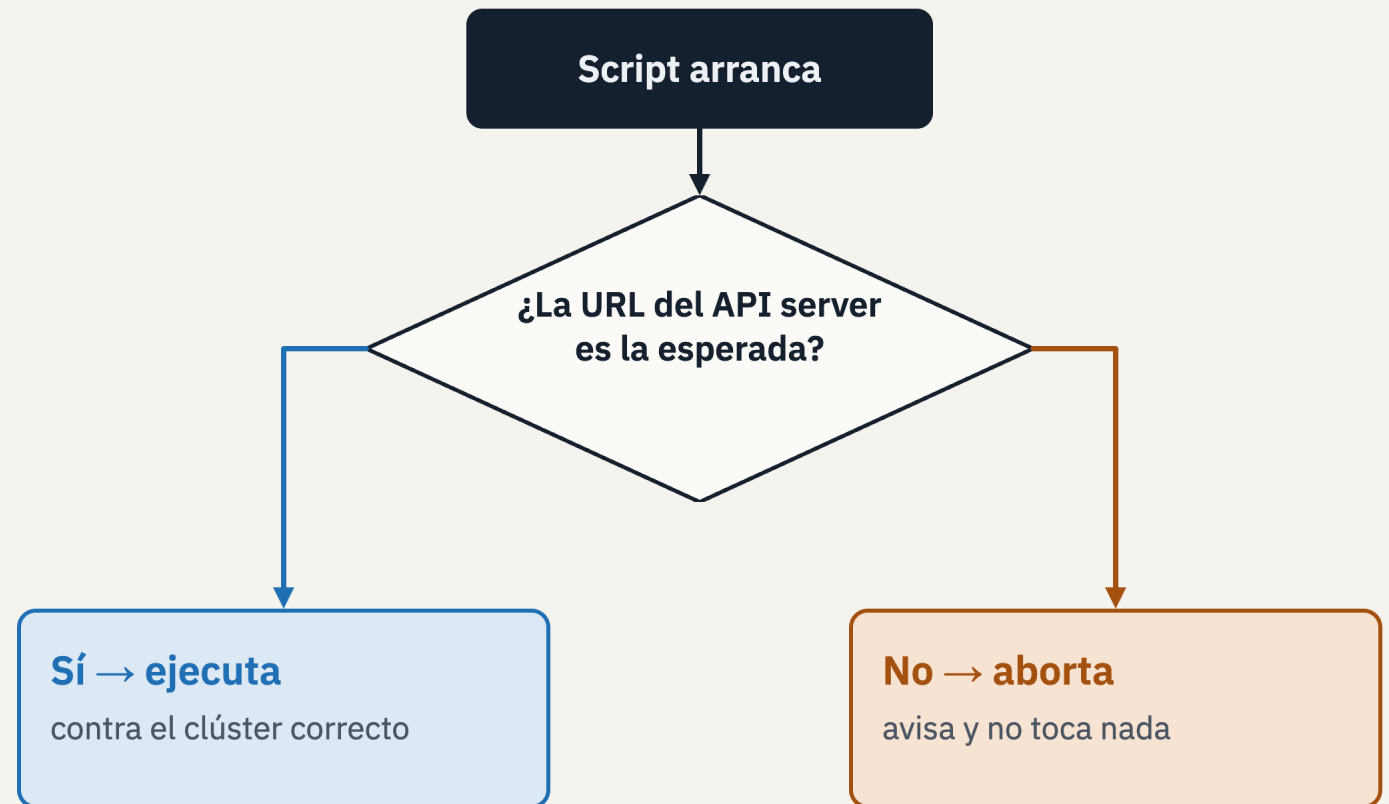
Cada script comprueba a qué clúster apunta

POR QUÉ

Laboratorio y producción comparten herramientas. Equivocarse de clúster es el error más barato de cometer y el más caro de pagar.

CÓMO

- Se compara la URL del API server, no el nombre del contexto
- El contexto, siempre visible en el prompt
- Producción nunca es el contexto por defecto
- Un script no se edita mientras se ejecuta



Mínimo privilegio en cada pieza, no solo en las personas

Identidad	Puede	No puede
Usuario RouterOS de lectura	read, rest-api, api	cambiar configuración
Usuario RouterOS de CI	aplicar cambios aprobados	usarse fuera de la CI
Robot de Harbor del tenant	pull de sus imágenes	push ni ver otros proyectos
Agente	leer su tenant y abrir PR	aprobar ni aplicar
Servidor MCP	resúmenes de solo lectura	devolver config en bruto
Persona	revisar y aprobar	aplicar sin pasar por CI

Por qué: el radio de impacto de una credencial robada es lo que esa credencial puede hacer.

ENS: op.acc, segregación de funciones y mínimo privilegio.

La CI no ejecuta nada que no esté fijado y firmado

POR QUÉ

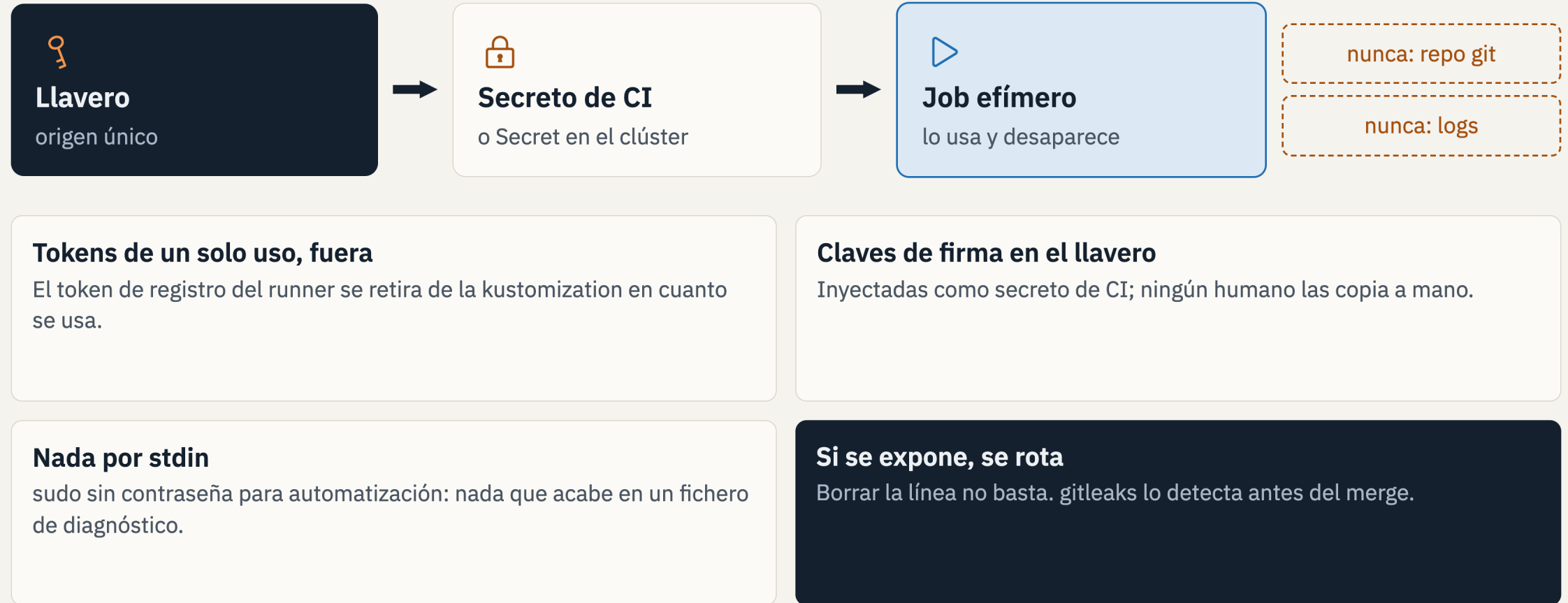
El runner tiene credenciales de registro y de firma. Una acción remota que cambia por debajo es un tercero ejecutando código en tu CI.

CÓMO

- Versión y hash de cada binario, en el repo
- gitleaks y Trivy en cada cambio
- Firma por digest, nunca por tag
- Versión de Kyverno fijada y probada



Los secretos no viven en git ni en ficheros



Copia continua de cada base de datos, y réplica antes de tocarla

COPIA CONTINUA

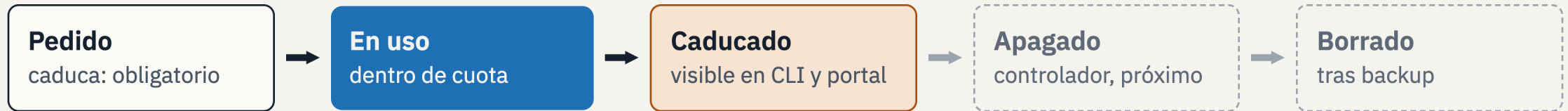


- Prueba de restauración periódica
- Copia fuera del sitio para desastre

ANTES DE UN CAMBIO QUE REINICIA



Todo recurso nace con fecha de caducidad



POR QUÉ

Los recursos olvidados se comen GPU y disco. En la nube pública lo paga la factura; aquí lo paga el vecino de tenant.

CÓMO

Campo caduca en las XRD de VM y de entorno, CuotaGPU como techo por tenant y laboratorios efímeros solo con fecha.

Cada cambio, de persona o de agente, deja evidencia



Pull request

qué cambió, quién lo propuso y quién lo aprobó



Audit log

quién llamó a la API del clúster, y con qué



Langfuse

qué hizo cada agente y con qué herramienta



Loki

qué pasó en nodos, pods y routers

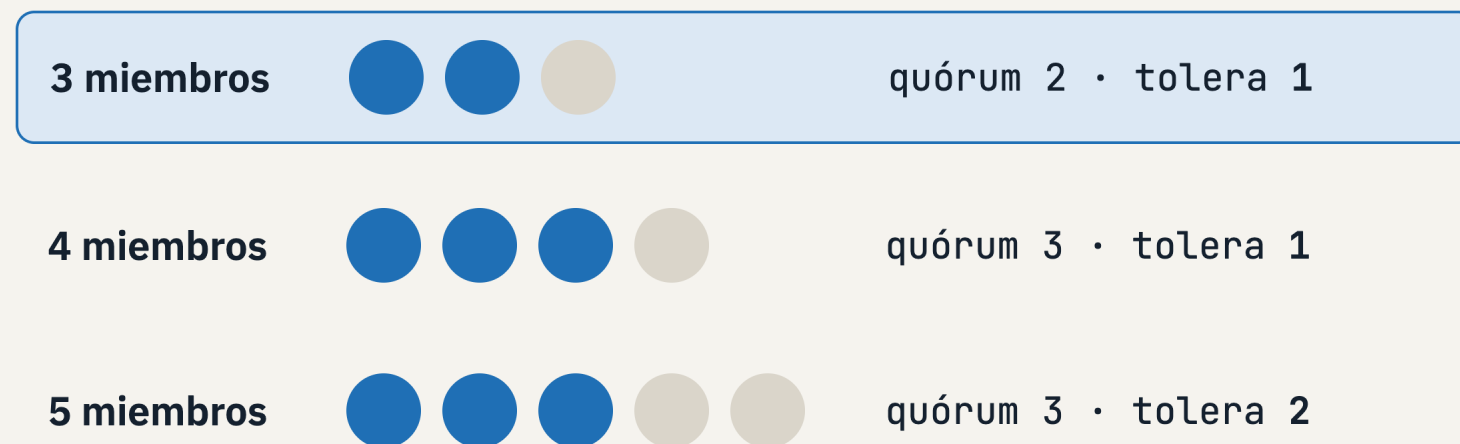


OpenCost

cuánto costó, por tenant

ENS op.exp: gestión de cambios y registro de actividad salen de la operación diaria, no de un informe aparte.

etcd impar, en servidores dedicados sin carga



azul = votos necesarios para el quórum · gris = miembros que pueden caer



En el clúster

Tres servidores dedicados sin GPU para el plano de control. Los nodos GPU, solo como agentes.

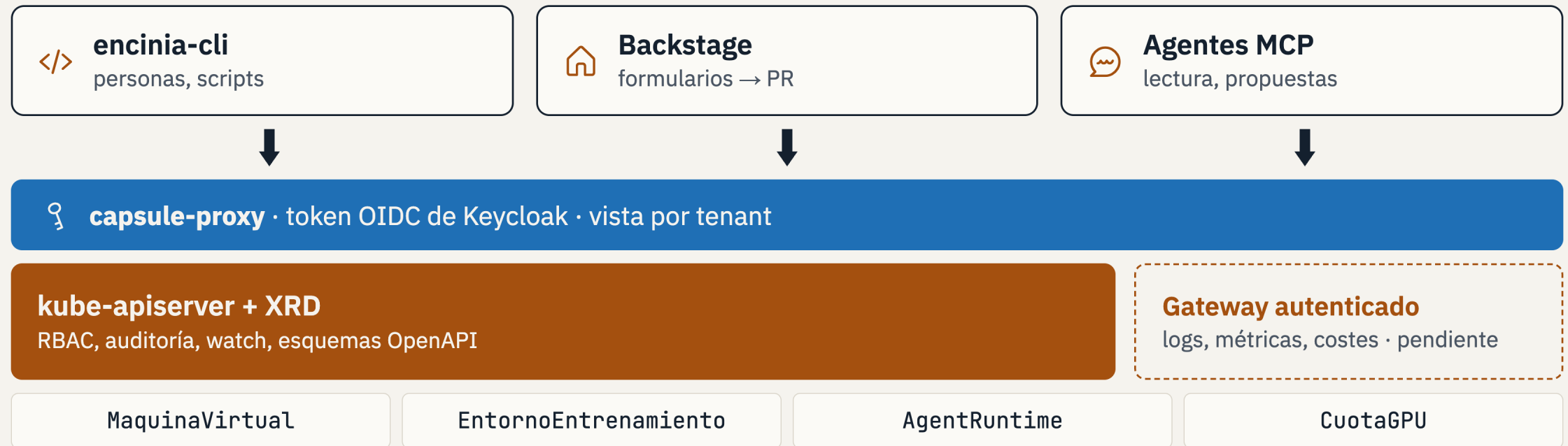
PARTE 2

02

API, CLI y dashboard

Un modelo de recursos, tres formas de usarlo.

Una API, tres clientes, los mismos permisos



Por qué no escribimos una API propia

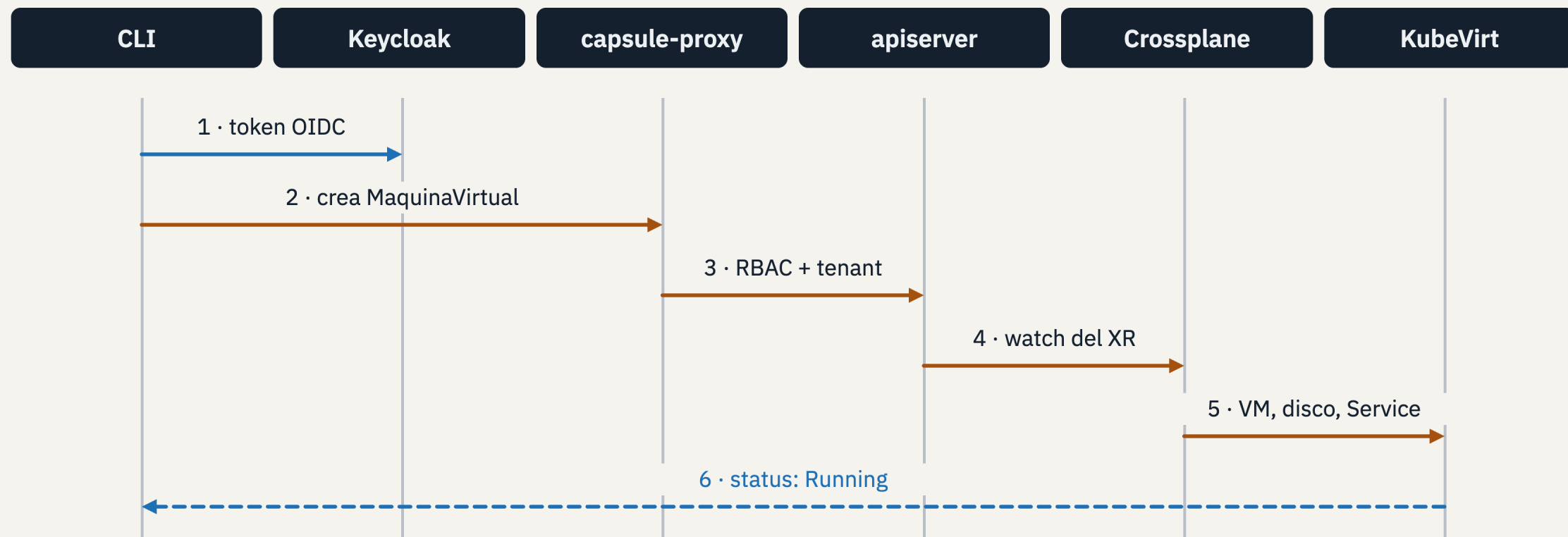
Necesidad	API REST propia	Kubernetes + XRD
Autenticación	a construir	OIDC del apiserver
Autorización	a construir	RBAC + Capsule
Auditoría	a construir	audit log del apiserver
Validación	código a mano	esquema OpenAPI de la XRD
Eventos	websockets propios	watch nativo
Clientes	SDK por lenguaje	kubectl, client-go, cualquier cliente
Coste real	servidor que mantener	esquemas versionados que diseñar bien

El trabajo no desaparece: pasa de escribir endpoints a diseñar contratos.

Cada servicio, un tipo de recurso

Servicio	Recurso que pides	Qué resuelve
 Máquina virtual	<code>MaquinaVirtual</code>	VM bajo demanda
 Entorno con GPU	<code>EntornoEntrenamiento</code>	entrenar y ajustar modelos
 Agente	<code>AgentRuntime</code>	desplegar agentes
 Cuota de GPU	<code>CuotaGPU</code>	reparto por tenant
 Bucket	<code>API S3 de RGW</code>	la excepción: protocolo estándar

De «quiero una VM» a Running, en seis mensajes



El esquema es el contrato público

```
kind: CompositeResourceDefinition
spec:
  scope: Namespaced
  versions:
  - name: v1alpha1
    schema: openAPIV3Schema:
      spec:
        imagen: {enum: [debian-13, ...]}
        tamano: {default: pequena}
        caduca: {format: date}
      status:
        fase, ipInterna, caducada
```

Mínimo y con defaults

Nada de KubeVirt asomando al usuario

status para personas

Fase, IP interna y caducidad legibles

Versionado explícito

v1alpha1 → v1beta1, una Composition por versión

Prueba de fuego: si hace falta leer la documentación de KubeVirt para pedir una VM, la XRD está mal.

Objetivo: cubrir el 99 % de los casos de uso que necesita el cliente local



Servicio, operación, opciones: un patrón conocido

```
$ encinia --help
uso: encinia <servicio> <operación> [opciones]
```

servicios

login · whoami	identidad OIDC
tenancy · quotas	organización
compute	máquinas virtuales
storage	almacenamiento de objetos
models	modelos
agents · mcp	agentes

globales

```
--output --query --profile --region
```

tarea → comando

listar mis máquinas virtuales

```
encinia compute list
```

listar mis objetos almacenados

```
encinia storage ls
```

saber con qué identidad trabajo

```
encinia whoami
```

ver los modelos disponibles

```
encinia models list
```

Lo que tomamos de las CLIs de nube y de kubectl

De las CLIs de nube

- --output json · yaml · table · text
- --query con JMESPath
- --profile y --region
- JSON por defecto para scripts

De kubectl y Krew

- --kubeconfig, --context, -n
- Nombres en kebab-case
- También como kubectl encinia
- Nada nuevo que aprender

```
$ encinia compute list --output table \  
  --query "[?fase=='Running' ].{vm:nombre,caduca:caduca}"
```

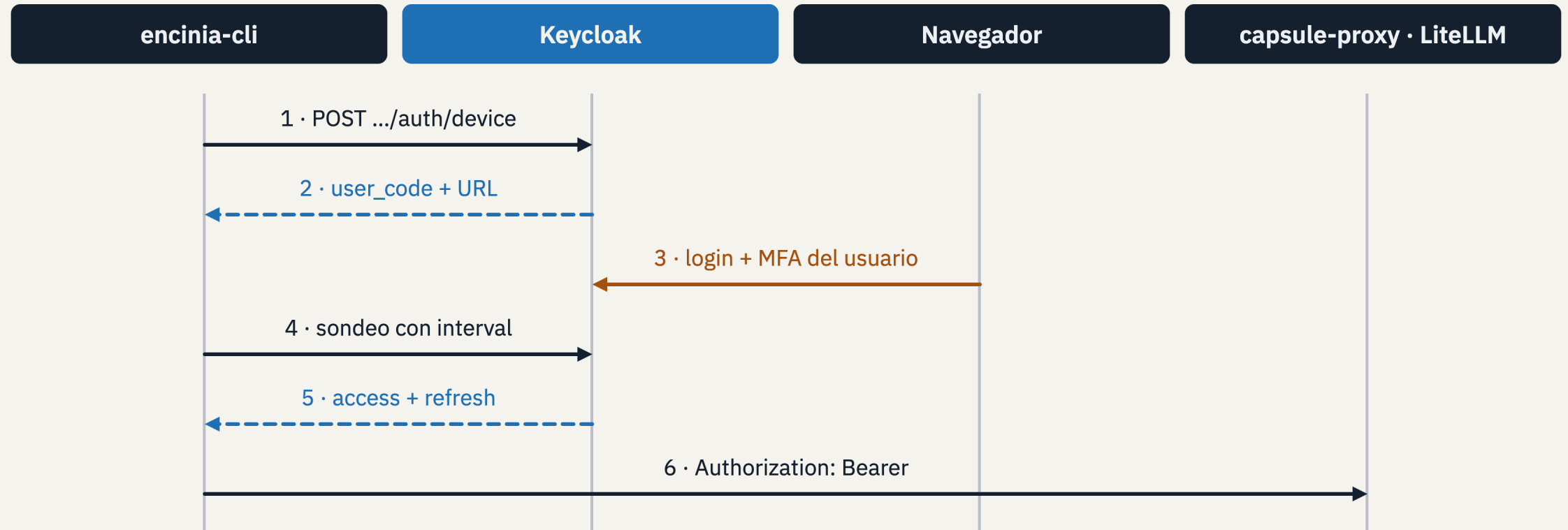
VM	CADUCA
lab-borde	2026-10-24
lab-acceso	2026-10-24

Un documento de sitio; --region elige el clúster



Mismo issuer OIDC en todas las regiones: un login sirve para los dos clústeres.

Device flow: la contraseña nunca toca la terminal



Una VM en tres comandos

```
$ encinia login
sesión iniciada · tenant operador001

$ encinia compute create lab-prueba \
  --image debian-13 --size small --expires 72h
maquinavirtual/lab-prueba creada

$ encinia compute list --output table
NOMBRE      IMAGEN      FASE      CADUCA
lab-prueba  debian-13   Running   72 h
```

detrás, en el namespace del tenant

MaquinaVirtual (XR)

VirtualMachine

DataVolume en Ceph

Service + NetworkPolicy

La tabla es el status del XR: la CLI no inventa nada.

Cada tenant ve sus modelos, sus agentes y su presupuesto

```
$ encinia models list --output table
```

ALIAS	USO	ESTADO
chat	general	activo
agentes	tools	activo
embeddings	RAG	a demanda

```
$ encinia mcp list
```

SERVIDOR	HERRAMIENTAS	MODOS
routers	routers, resumen	lectura
ceph	estado, pools	lectura



Claves virtuales

Una por equipo, con presupuesto y límites en LiteLLM



Aislamiento

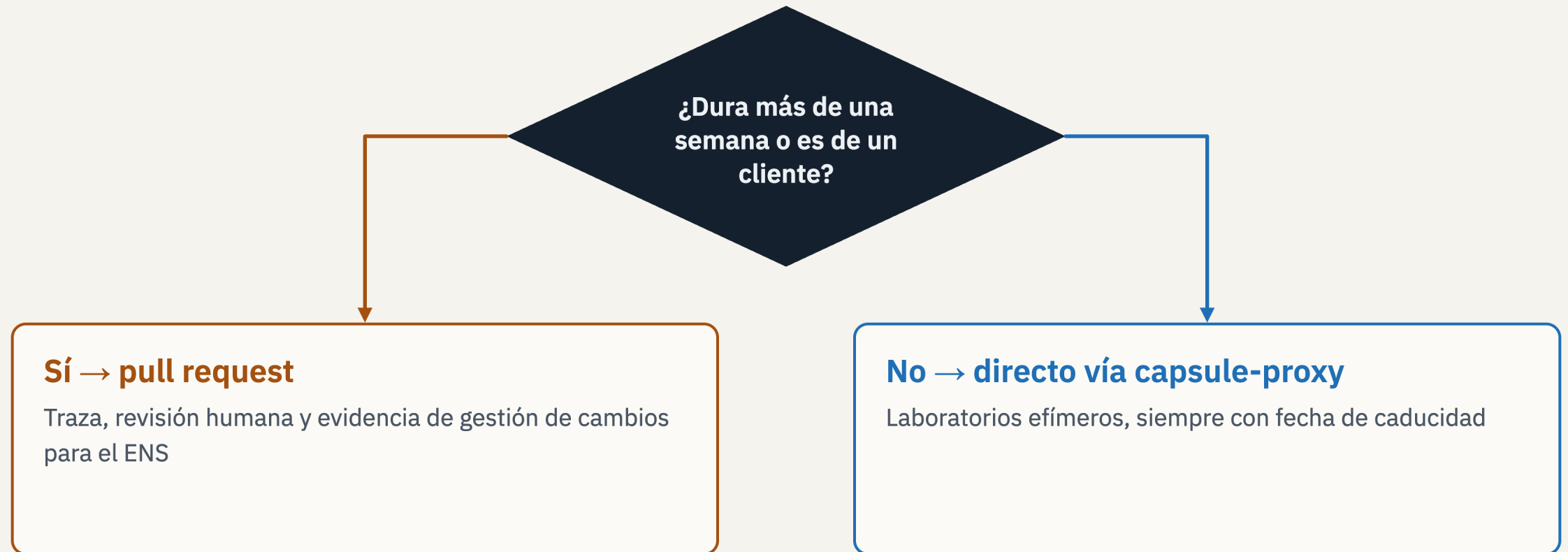
Un tenant no ve los modelos ni gasta el presupuesto de otro

El formulario no llama a una API mágica: abre un PR



Patrón conocido: el plugin Kubernetes Ingestor de TeraSky genera una plantilla por cada XRD del clúster.

¿PR o aplicación directa?



Seis paneles, un mismo login



Backstage

Catálogo y plantillas

catálogo de servicios



Headlamp

Kubernetes por tenant, con Crossplane y KubeVirt

consola de Kubernetes



Grafana

Métricas, logs y GPU por tenant

monitorización



Langfuse

Trazas y evaluación de agentes

trazas de LLM



OpenCost

Coste por namespace y tenant

análisis de costes



LibreChat

Chat para usuarios finales contra LiteLLM

chat para usuarios

PARTE 3

03

Tres casos de uso

A

Laboratorio de operador

B

Entornos GPU para docentes e investigadores

C

Plataforma agéntica

Un operador entero en un tenant

Lo que cualquier operador querría para probar cambios antes de tocar la red real.

en marcha

20 routers CHR 7.24.5 con enlaces L2 para una clase de Loopback 0, y servidor MCP verificado de punta a punta

lab

VMs CHR + NAD L2

red

servidor mcp-routeros

agentes

auditor, cambios, bastionado

monitoring

syslog y SNMPv3 del tenant

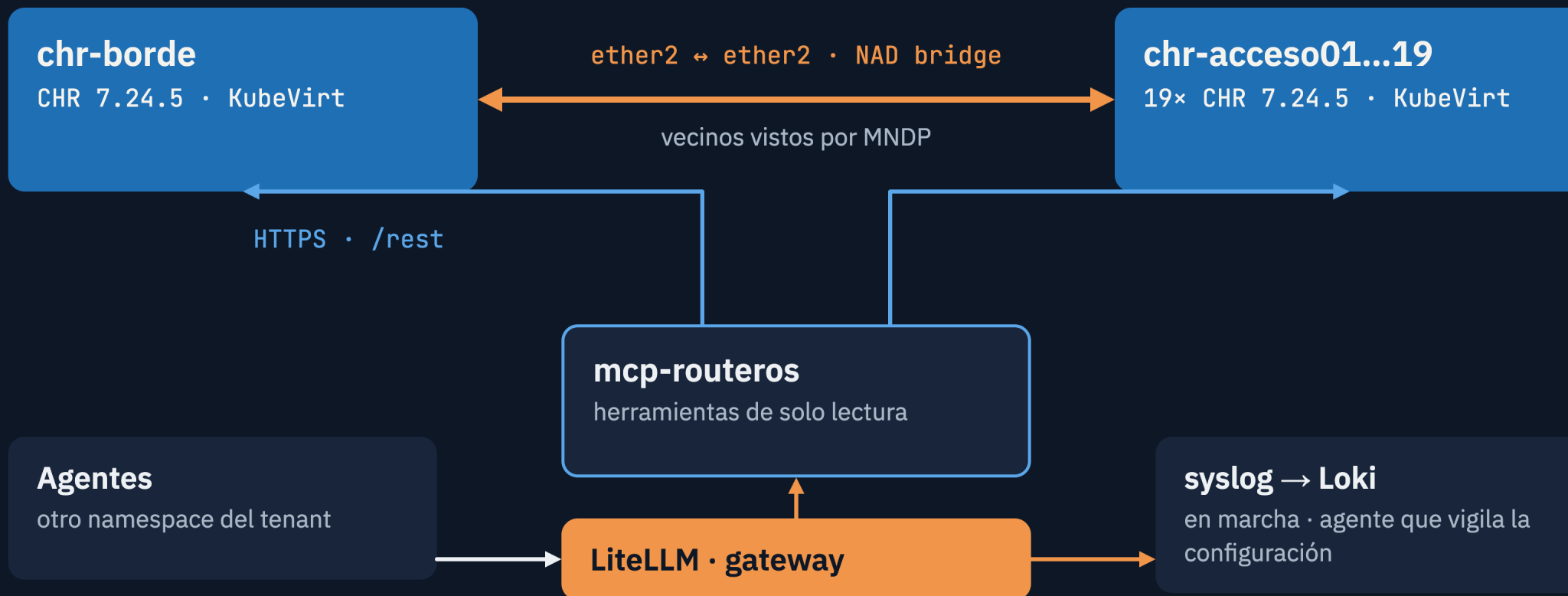
radius

FreeRADIUS para AAA

Tenant operador001

cuotas, políticas y RBAC comunes

Veinte routers, enlaces L2 reales, gestión solo por API



La consola, una vez. Después, solo API.

fase 0 • guest-agent

Solo habilitar www-ssl con certificado de la CA interna

fase 1 • REST API, para siempre

Usuarios, grupos, servicios, firewall, syslog, SNMPv3, NTP... todo declarativo en git

Usuarios que crea el bootstrap

Usuario	Políticas	Uso
lectura	read, rest-api, api	MCP, monitorización
ci	escritura limitada	solo desde CI



Guarda de contexto

El script aborta si el apiserver no es el del clúster de laboratorio (buena práctica 03).

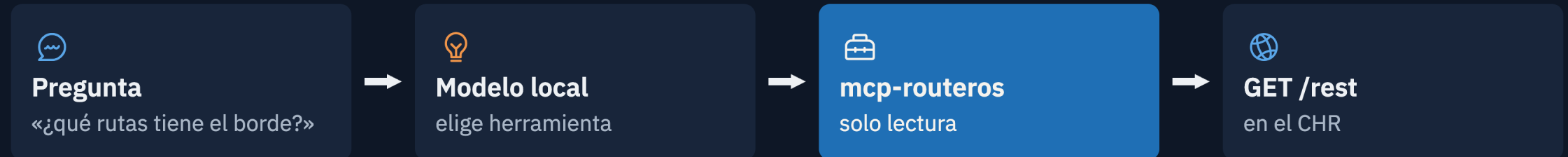
Verbos HTTP sobre comandos de siempre

GET	print
PUT	add
PATCH	set
DELETE	remove
POST	cualquier comando

```
$ curl -s --cacert ca.pem -u lectura:** \
https://<chr>/rest/system/resource
{"uptime":"2d4h","cpu-load":"3",
 "version":"7.24.5 (stable)", ...}
```

- Los valores JSON vuelven siempre como string
- Máximo 60 s por petición; nada de comandos continuos
- Basic Auth, por eso solo HTTPS

El agente pregunta; la herramienta resume



```
llamada MCP (JSON-RPC 2.0)
{"method": "tools/call",
 "params": {"name": "resumen",
 "arguments": {"router": "chr-borde"}}}
```

Nunca devuelve la configuración en bruto

Sin credenciales de escritura

Cada llamada, trazada en Langfuse

Del RD 311/2022 a un módulo OpenTofu

Medida ENS	Control en RouterOS	Evidencia
op.acc · acceso	Usuarios por rol, grupos mínimos, solo HTTPS y SSH	Export de usuarios y servicios
op.exp · explotación	Config en git, cambios por PR, NTP, syslog a Loki	Historial de PR y logs
mp.com · comunicaciones	Input con drop por defecto, SNMPv3	Reglas de firewall

op.acc.4

Todo acceso prohibido salvo autorización expresa

mp.com.1

Flujos por el perímetro, autorizados previamente

traducción

chain=input action=drop al final, y nada más

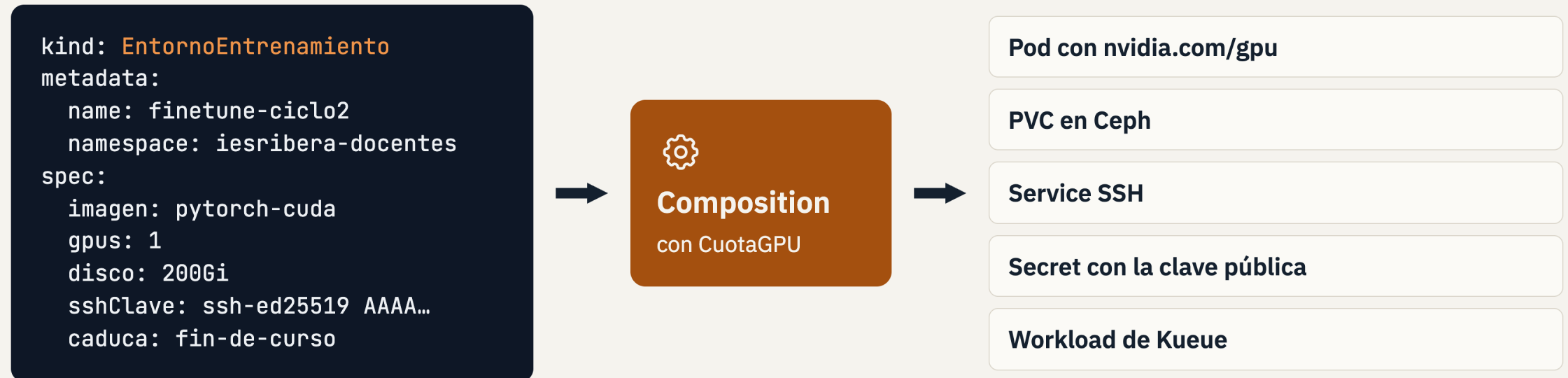
El agente audita y propone; una persona aprueba



Enseñar e investigar con modelos propios, sin salir del CPD

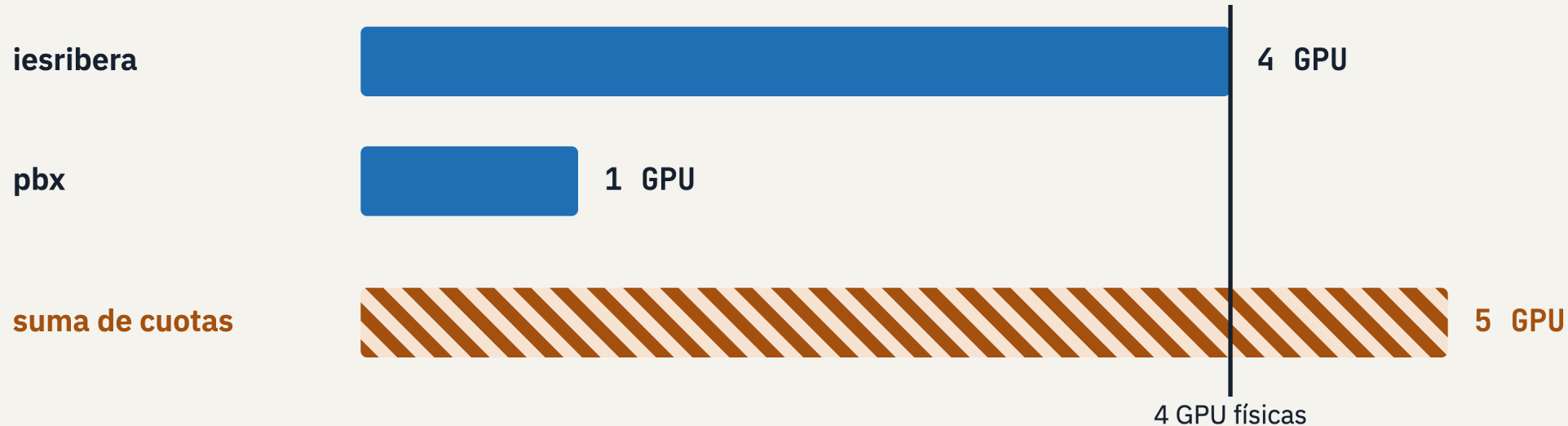


Para quien enseña o investiga, una máquina; debajo, cinco objetos



\$ encinia training list • un entorno de entrenamiento gestionado, en el CPD

La cuota es un techo, no una reserva



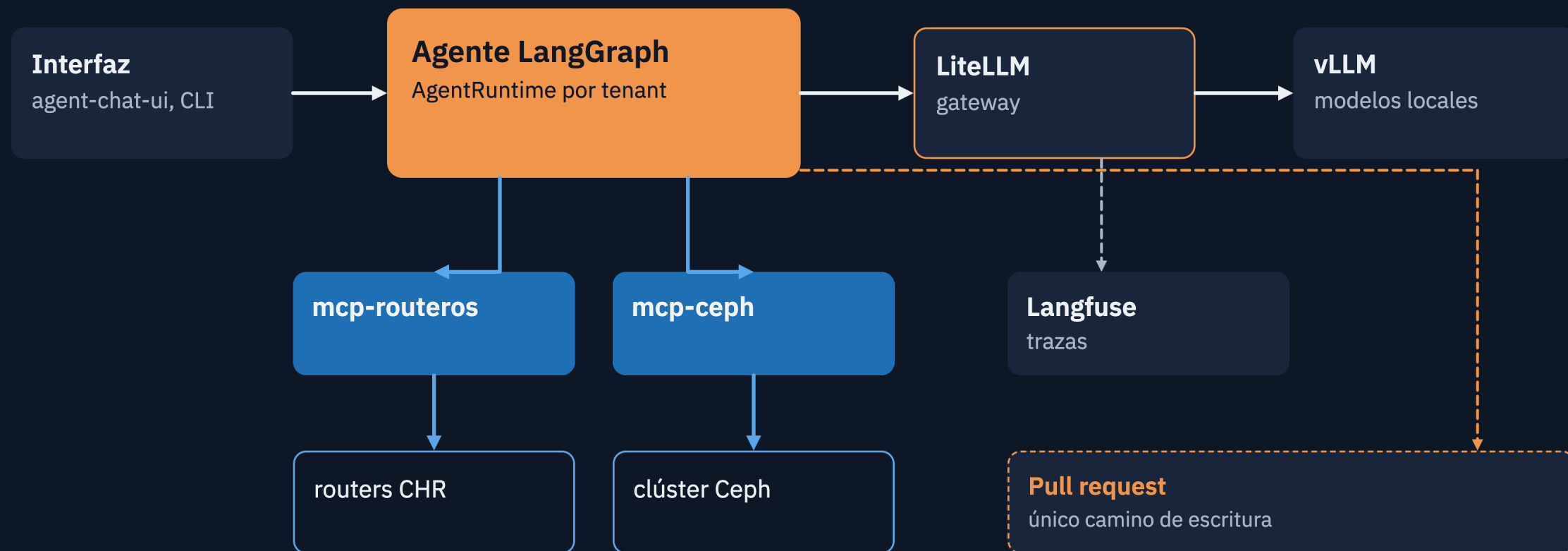
Capsule + Kueue

ResourceQuota del tenant y ClusterQueue con nominalQuota

Consecuencia

Sin colas, las cuotas no evitan que alguien se quede esperando

Una plataforma de agentes con modelos que no salen del CPD



Un agente es una aplicación más, con sus límites

```
kind: AgentRuntime
metadata:
  name: red-auditor
  namespace: operador001-agentes
spec:
  imagen: <registro>/red-auditor
  modelo: agentes
  herramientas:
    - mcp-routeros
  presupuesto: 20 EUR/mes
```



Composition



Deployment firmado

Clave virtual de LiteLLM

Proyecto en Langfuse

Egress solo a gateway y MCP

ServiceAccount de lectura

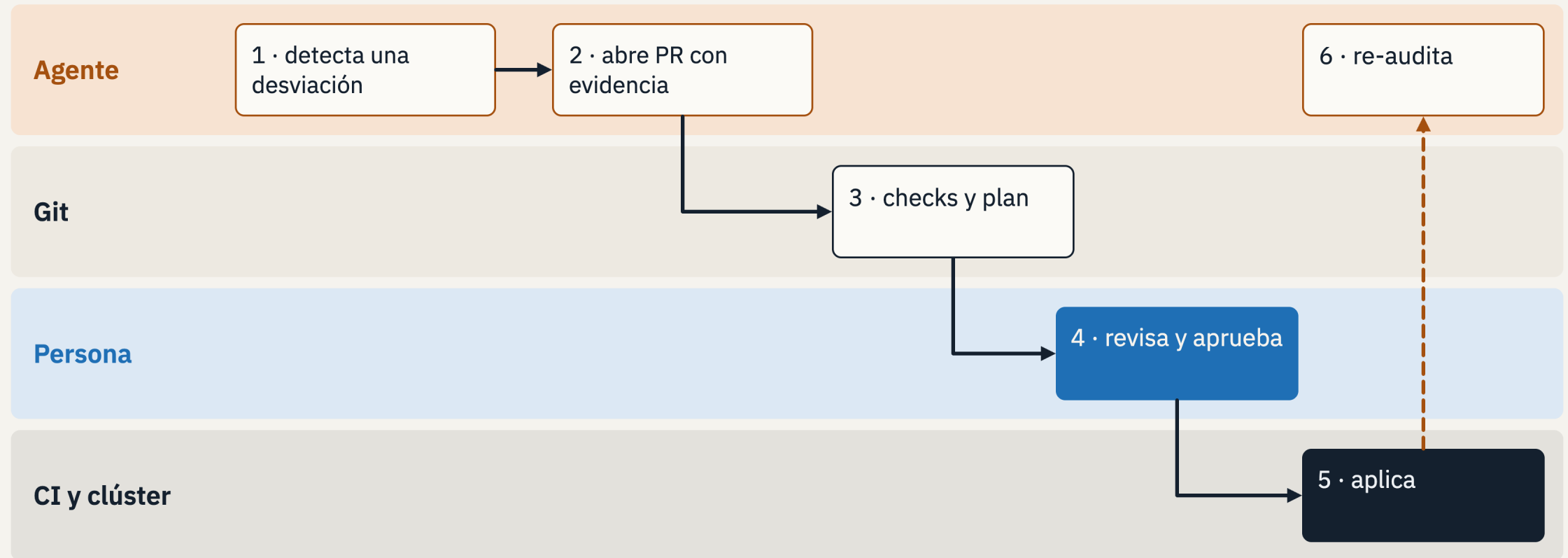
El agente propone; no tiene la llave

Acción	Agente	Persona	CI
Leer Kubernetes del tenant	sí	sí	sí
Consultar routers por MCP	sí	sí	no aplica
Abrir un pull request	sí	sí	no aplica
Aprobar el pull request	no	sí	no
Aplicar en producción	no	no, vía CI	sí

op.acc • mínimo privilegio

op.exp • gestión de cambios

Cuatro carriles, una sola puerta de escritura



Sin trazas, un agente es una caja negra

traza • «¿qué rutas tiene el borde?»

agente



llm • planificar



mcp • resumen



GET /rest



llm • redactar



Coste

por ejecución, agente y tenant

Evaluación

regresiones al cambiar de modelo

Auditoría

qué herramienta, con qué datos

Lo que aprendimos con agentes

1

Modelos locales: suficientes para diagnóstico y auditoría

No para todo. Elegir tarea antes que modelo.

2

Herramientas estrechas y de lectura

Cuanto más concreta la herramienta, mejor se porta el agente.

3

Aprobación humana desde el diseño

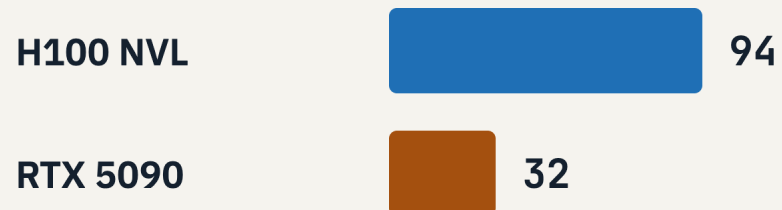
No es un parche que se añade al final.

Más memoria total, pero repartida en trozos de 32 GB

memoria de GPU total (GB)



memoria por GPU (GB)



Blackwell

Driver open ≥ 570 y CUDA ≥ 12.8 ; FP8 y NVFP4

Sin NVLink

TP solo por PCIe: modelos de 1 GPU o TP=2 en el mismo socket

CONCLUSIONES

Cuatro ideas para llevarse

1 Una API, tres clientes. Kubernetes + XRD ya es la API de servicios.

2 La red, también por API. CHR, REST y MCP dentro de la plataforma.

3 ENS como código. Evidencia generada y cambios por PR.

4 Prácticas de operador. Cambios escalonados, mínimo privilegio y todo trazado.

ESNOG 2026 • BARCELONA

¿Preguntas?

GRACIAS A

Oliver Garrido

Gabriel Roque

Iván López

Manuel de Castro

José Manuel Román

Fibercli • Encinia

Cualquiera de las prácticas, con más detalle, en
el pasillo.

Las marcas y nombres de productos citados pertenecen a sus titulares y se mencionan solo para identificar el software utilizado. «S3» se usa como nombre del protocolo de almacenamiento de objetos. No implica relación ni respaldo de ningún fabricante o proveedor.

ENLACES Y CONTACTO

Gracias



Formación

Loopback 0

Cursos y laboratorios para ingenieros de red, como el de los 20 CHR.

lo0.es



Consultoría

Fibercli

Red, ciberseguridad, ENS y plataformas soberanas para operadores.

fibercli.com



Plataforma agéntica
• B2B

Encinia

IA privada, agentes y cómputo en las instalaciones del cliente.

encinia.ai



Ecuador

Gigaipnet

Hosting, servidores y servicios cloud en Ecuador.

gigaipnet.com

José Manuel Román [`jose.roman@fibercli.com`](mailto:jose.roman@fibercli.com)