

Desplegar con Safent

El clúster se conecta **una vez, con un comando**. Desde entonces, cada versión de cada app llega y se instala sola. Safent no entra nunca en el clúster.

Versión al día: docs.lab.safentmcp.com/despliegue

Conectar el clúster una vez

1 Ten un clúster con Kubernetes 1.34 o posterior

Si ya lo tienes, pasa al 2. Si no, en un servidor Linux con Internet es un comando, como root (tarda un minuto):

```
curl -sfL https://get.k3s.io | sh -
```

En Azure, AWS, Google Cloud o HPE PCAI: [Crear el clúster](#).

✓ Listo si `kubectl get nodes` muestra los nodos en `Ready`.

2 Da de alta el entorno en Safent

En equipo.lab.safentmcp.com: **Cartera** › el cliente › el proyecto › pestaña **Entornos** › **Añadir un entorno**. Pon un nombre (`planta-norte`) y el tipo, y pulsa **Dar de alta**. Si el clúster es tuyo y sirve a todos tus clientes, entra por **Infraestructura del partner**, arriba en la Cartera.

✓ Listo si aparece el comando de conexión. Cópialo: solo se enseña una vez.

3 Pega el comando en un terminal con `kubectl`

En el ordenador desde el que administras el clúster (en k3s, en el propio servidor, después de `sudo -i`). Tarda menos de un minuto.

✓ Listo si termina en «Listo» y, en Safent, el entorno sale **Conectado**.

Instalar una app una vez por app

1

Elige dónde se instala

En Safent: el proyecto › **Repositorios** › el repositorio › **Despliegue** › **Elegir dónde se despliega** › el entorno › **Fijar destino**. Preproducción recibe cada versión; producción, desde que se acepta la UAT.

✓ **Listo si la línea de tiempo del destino llega a Instalada.**

2

Si se para en «Falta la configuración», ponla

Son los datos que la app necesita del cliente (su base de datos, sus claves...). En Safent, **Configuración de la app** enseña qué variables pide y el comando, ya con su nombre. Escribe los valores en `<app>.env`, una línea `NOMBRE=valor` cada una, sin comillas, y ejecuta el comando en el terminal con `kubectl`:

```
echo "_huella=$(date +%s)" | kubectl -n flux-system create secret generic
<app>-config --from-env-file=<app>.env --from-env-file=/dev/stdin --dry-
run=client -o yaml | kubectl label --local -f -
reconcile.fluxcd.io/watch=Enabled -o yaml | kubectl apply -f -
```

Los valores se quedan en el clúster: Safent solo ve los nombres. Para cambiar uno, edita el fichero y repite el comando.

✓ **Listo si en menos de un minuto pasa a Instalada.**

Y ya está. Las versiones siguientes llegan solas, y si una no arranca, el clúster vuelve solo a la anterior.

SI LO NECESITAS

Crear el clúster en Azure, AWS, Google Cloud o HPE PCAI

Comandos de cada proveedor, con `kubectl` de administrador al final.

Hace falta Kubernetes 1.34 o posterior (x86 o ARM), `kubectl` con un usuario administrador y un almacenamiento por defecto (k3s, AKS y GKE ya lo traen). Si las apps se publican con URL propia, también un dominio, un ingress y certificados.

Servidor Linux (k3s)

Probado por Safent

1. Entra en el servidor, por ejemplo con `ssh tu-usuario@tu-servidor` . Tiene que ser Linux (Ubuntu, Debian o RHEL), x86 o ARM, con al menos 2 CPU, 4 GB de memoria y salida a Internet por HTTPS.
2. Instala Kubernetes (k3s) como root. Tarda un minuto y trae `kubectl` :

```
curl -sL https://get.k3s.io | sh -
```

3. Hazte root para el resto de pasos:

```
sudo -i
```

4. Comprueba que el nodo está listo (STATUS `Ready` y VERSION v1.34 o posterior):

```
kubectl get nodes
```

Para borrarlo todo al terminar las pruebas: `/usr/local/bin/k3s-uninstall.sh`

Azure (AKS)

Según la documentación oficial de Microsoft

1. Instala [Azure CLI](#) e inicia sesión (se abre el navegador):

```
az login
```

2. Instala `kubectl` (en Linux o macOS, con `sudo` delante si te lo pide):

```
az aks install-cli
```

3. Crea un grupo de recursos:

```
az group create --name safent --location westeurope
```

4. Crea el clúster. Usa AKS estándar, no Automatic: sus salvaguardas pueden rechazar despliegues. Tarda unos minutos.

```
az aks create --resource-group safent --name safent --node-count 2 --node-vm-size Standard_D4s_v5 --generate-ssh-keys
```

5. Trae las credenciales a `kubectl` :

```
az aks get-credentials --resource-group safent --name safent
```

6. Comprueba la versión («Server Version» v1.34 o posterior):

```
kubectl version
```

Si el clúster usa Entra ID con Azure RBAC, el usuario que conecta necesita el rol «Azure Kubernetes Service RBAC Cluster Admin». Para borrarlo: `az group delete --name safent --yes --no-wait`

AWS (EKS)

Según la documentación oficial de AWS

1. Instala [AWS CLI](#), [eksctl](#) y [kubectl](#), y configura las credenciales con `aws configure` (o `aws sso login`, si la empresa usa SSO). Comprueba que responde con la cuenta buena:

```
aws sts get-caller-identity
```

2. Crea el fichero `cluster.yaml` :

```
cat > cluster.yaml <<'EOF'
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig
metadata: { name: safent, region: eu-west-1 }
autoModeConfig: { enabled: true }
EOF
```

3. Crea el clúster (EKS Auto Mode). Tarda unos 15 minutos y deja `kubectl` apuntando a él:

```
eksctl create cluster -f cluster.yaml
```

4. Marca un almacenamiento por defecto (Auto Mode no crea ninguno):

```
kubectl apply -f - <<'EOF'
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: auto-ebs-sc
  annotations: { storageclass.kubernetes.io/is-default-class: "true" }
allowedTopologies:
  - matchLabelExpressions:
    - { key: eks.amazonaws.com/compute-type, values: [auto] }
provisioner: ebs.csi.eks.amazonaws.com
volumeBindingMode: WaitForFirstConsumer
parameters: { type: gp3, encrypted: "true" }
EOF
```

5. Comprueba la versión («Server Version» v1.34 o posterior):

```
kubectl version
```

Solo quien crea el clúster es administrador. Si el comando de conexión lo ejecuta otra persona, dale acceso con `aws eks create-access-entry` y la política `AmazonEKSClusterAdminPolicy`. Para borrarlo: `eksctl delete cluster -f cluster.yaml --wait`

Google Cloud (GKE)

Según la documentación oficial de Google

1. Instala [Google Cloud CLI](#), inicia sesión y elige el proyecto (cambia `TU-PROYECTO` por su ID):

```
gcloud auth login && gcloud config set project TU-PROYECTO
```

2. Activa la API de Kubernetes e instala `kubectl` y el complemento de acceso:

```
gcloud services enable container.googleapis.com && gcloud components install kubectl gke-gcloud-auth-plugin
```

Si instalaste gcloud con apt, esos dos van con `sudo apt-get install kubectl google-cloud-cli-gke-gcloud-auth-plugin`.

3. Crea el clúster (GKE Standard; Autopilot no está comprobado con Flux):

```
gcloud container clusters create safent --location=europe-southwest1-a --num-nodes=2 --machine-type=e2-standard-4
```

4. Trae las credenciales a `kubectl`:

```
gcloud container clusters get-credentials safent --location=europe-southwest1-a
```

5. Comprueba la versión («Server Version» v1.34 o posterior):

```
kubectl version
```

Para borrarlo: `gcloud container clusters delete safent --location=europe-southwest1-a`

HPE Private Cloud AI

Por validar contigo en la primera PCAI

1. Una versión de PCAI que traiga Kubernetes 1.34 o posterior. Según la matriz de soporte de HPE, AIE 2026.07 trae la 1.34.2; las anteriores no llegan.
2. Pide al administrador de PCAI un kubeconfig con permisos de administrador del clúster y úsalo:

```
export KUBECONFIG=$HOME/pcai-admin.kubeconfig
```

3. Comprueba que eres administrador (tiene que responder **yes**):

```
kubectl auth can-i create clusterrolebindings
```

4. Comprueba la versión («Server Version» v1.34 o posterior):

```
kubectl version
```

HPE no documenta públicamente cómo obtener ese kubeconfig de administrador ni si PCAI admite herramientas como Flux, que instala sus propios recursos en el clúster. Es lo primero que comprobaremos contigo en la primera PCAI que se conecte. Si no da un administrador del clúster o no tiene salida a Internet, se usa el [modo sin Internet](#): cada versión llega en un paquete firmado que se instala con Helm.

Si la red del clúster filtra la salida a Internet

Lo único que hay que abrir: HTTPS de salida a dos direcciones.

Pide a quien lleve la red que permita, desde los nodos del clúster, la **salida por HTTPS (TCP 443)** a

`forgejo.lab.safentmcp.com` y `equipo.lab.safentmcp.com`, por nombre de dominio. No se abre nada de entrada. Compruébalo desde el terminal del paso 3: la primera tiene que dar **200** y la segunda **401** (el registro responde y pide la credencial que pone el comando).

```
curl -sS -o /dev/null -w '%{http_code}\n' https://equipo.lab.safentmcp.com/salud
curl -sS -o /dev/null -w '%{http_code}\n' https://forgejo.lab.safentmcp.com/v2/
```

Si el clúster sale a Internet por un proxy

Cuatro cosas antes de pegar el comando; lo demás lo hace él.

1. **En Safent:** ficha del entorno › **Editar** › **Proxy de salida:** `http://proxy.empresa.local:3128` › **Guardar**. Si pide usuario y contraseña, van en la dirección (`http://usuario:clave@proxy.empresa.local:3128`); Safent no vuelve a enseñar la clave.
2. **En el proxy:** deja salir a `forgejo.lab.safentmcp.com` y `equipo.lab.safentmcp.com` (443). Desde un nodo tiene que dar **401** :

```
curl -sS -x http://proxy.empresa.local:3128 -o /dev/null -w '%{http_code}\n'
https://forgejo.lab.safentmcp.com/v2/
```

3. **En los nodos:** que descarguen las imágenes por el proxy (si ya descargan imágenes de Internet, no hay que tocar nada):

k3s

Probado por Safent

1. En el servidor, como root, añade el proxy al entorno de k3s (en los nodos agente, el fichero es `/etc/systemd/system/k3s-agent.service.env`):

```
cat >> /etc/systemd/system/k3s.service.env <<'EOF'
CONTAINERD_HTTP_PROXY=http://proxy.empresa.local:3128
CONTAINERD_HTTPS_PROXY=http://proxy.empresa.local:3128
CONTAINERD_NO_PROXY=127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16
EOF
```

2. Reinicia k3s (en los agentes, `systemctl restart k3s-agent`). Las apps que ya corren no se paran:

```
systemctl restart k3s
```

Kubernetes con containerd (kubeadm y similares)

Configuración estándar de systemd

1. En cada nodo, como root:

```
mkdir -p /etc/systemd/system/containerd.service.d
cat > /etc/systemd/system/containerd.service.d/http-proxy.conf <<'EOF'
[Service]
Environment="HTTP_PROXY=http://proxy.empresa.local:3128"
Environment="HTTPS_PROXY=http://proxy.empresa.local:3128"
Environment="NO_PROXY=127.0.0.0/8,10.0.0.0/8,172.16.0.0/12,192.168.0.0/16, .
svc, .cluster.local"
EOF
systemctl daemon-reload && systemctl restart containerd
```

2. Comprueba que lo tiene (tiene que salir el proxy):

```
systemctl show containerd --property=Environment
```

Azure (AKS)

Según la documentación oficial de Microsoft

1. Crea el fichero con el proxy:

```
cat > aks-proxy.json <<'EOF'
{ "httpProxy": "http://proxy.empresa.local:3128", "httpsProxy":
"http://proxy.empresa.local:3128", "noProxy": ["localhost", "127.0.0.1"] }
EOF
```

2. Aplícalo al clúster (con los nombres del clúster):

```
az aks update --resource-group safent --name safent --http-proxy-config
aks-proxy.json
```

AKS reinstala todos los nodos al cambiar el proxy (las apps se reinician) y no admite proxy con usuario y contraseña. Desde entonces pone el proxy a los nodos y a los pods.

AWS (EKS)

Según la documentación oficial de AWS

- **EKS Auto Mode** (el de «Crear el clúster»): el proxy va en el NodeClass de los nodos, en `advancedNetworking`. Cómo crear el NodeClass y el NodePool que lo usa: [guía de AWS](#).

```
advancedNetworking:
  httpsProxy: http://proxy.empresa.local:3128
  noProxy: [localhost, 127.0.0.1, 169.254.169.254, .internal,
.eks.amazonaws.com]
```

- **Grupos de nodos gestionados**: en los datos de usuario de su plantilla de lanzamiento, como en la [guía de AWS](#).

Google Cloud (GKE)

Según la documentación oficial de Google

GKE no admite proxy en los nodos. Si son privados, Google recomienda darles salida con Cloud NAT (y limitarla a las dos direcciones en el cortafuegos). Con su red y su región:

```
gcloud compute routers create safent-nat --network=default --region=europe-southwest1
gcloud compute routers nats create safent-nat --router=safent-nat --region=europe-southwest1 --auto-allocate-nat-external-ips --nat-all-subnet-ip-ranges
```

HPE Private Cloud AI

Según la documentación oficial de HPE

El proxy de los nodos se pone al instalar PCAI ([Setting up the proxy server](#)). Si ya lo tiene, basta con que el proxy deje salir a las dos direcciones (punto 2).

4. En el terminal, si tampoco sale directo a Internet (en k3s, el propio servidor), antes de pegar el comando:

```
export HTTPS_PROXY=http://proxy.empresa.local:3128
NO_PROXY="localhost,127.0.0.1,$(kubectl config view --minify -o jsonpath='{.clusters[0].cluster.server}' | sed 's#[a-z]*://##; s#[:/].*##')"
```

Si la red inspecciona el HTTPS

Proxy o cortafuegos con certificado propio: se pone en Safent y en los nodos.

Se nota porque al pegar el comando sale `curl: (60) SSL certificate problem`. Pide a sistemas el **certificado raíz** de la empresa: un fichero `.crt` o `.pem` que empieza por `-----BEGIN CERTIFICATE-----`.

1. En **Safent**: ficha del entorno › **Editar** › **Certificado de la empresa**: pega el certificado › **Guardar**. Flux y el agente confiarán en él.
2. En **cada nodo** y en el equipo donde pegas el comando, como root, con el certificado guardado en `empresa-ca.crt` (en Red Hat o Rocky: `cp empresa-ca.crt /etc/pki/ca-trust/source/anchors/ && update-ca-trust`):

```
cp empresa-ca.crt /usr/local/share/ca-certificates/empresa-ca.crt && update-ca-certificates
```

Después reinicia k3s (en los agentes, `k3s-agent`) o, con otro Kubernetes, `containerd`. Las apps que ya corren no se paran:

```
systemctl restart k3s
```

En AKS, el certificado va en el fichero del proxy: `"trustedCa": "<el certificado en base64>"`.

3. **Comprueba** desde un nodo que da `401` y no `curl: (60)` (con proxy, después del `export` del terminal):

```
curl -sS -o /dev/null -w '%{http_code}\n' https://forgejo.lab.safentmcp.com/v2/
```

Y pega el comando de conexión (o repítelo, si ya lo habías pegado).

Si el comando se para

Dice qué falta. Lo arreglas y lo vuelves a pegar: repetirlo no rompe nada.

MENSAJE	QUÉ HACER
<code>curl: (6)</code> o <code>curl: (7)</code> nada más pegar el comando	El terminal no llega a <code>equipo.lab.safentmcp.com</code> : salida a Internet . Si sale por un proxy, ponlo en el terminal (proxy , punto 4) y vuelve a pegar el comando.
<code>curl: (60)</code> nada más pegar el comando, o «este equipo no confía en su certificado»	La red inspecciona el HTTPS: certificado de la empresa , puntos 2 y 3, en este equipo.
Falta kubectl...	Instala Kubernetes (o k3s) y <code>kubectl</code> .
kubectl no llega al clúster	Revisa <code>KUBECONFIG</code> (en k3s, entra como root con <code>sudo -i</code>).
Hace falta un usuario administrador	Ejecútalo con un usuario <code>cluster-admin</code> .
⚠ Flux se soporta desde Kubernetes 1.34	El entorno se conecta, pero no está apto para instalar hasta que se actualice el clúster.
Este equipo no llega a ... / No se llega al registro de Safent	El ordenador donde ejecutas el comando no sale a esa dirección: salida a Internet . Si sale por un proxy, ponlo en el terminal (proxy , punto 4).
Flux no arranca	Si sus pods están en <code>ImagePullBackOff</code> (<code>kubectl -n flux-system get pods</code>), los nodos no salen a <code>forgejo.lab.safentmcp.com</code> aunque el ordenador desde el que se ejecuta sí: la salida desde los nodos. Con proxy, «En los nodos» .
El agente no arranca	Lo mismo, con <code>kubectl -n safent-entorno describe job/...</code> : los nodos no descargan su imagen de <code>forgejo.lab.safentmcp.com</code> .
El agente no pudo avisar a Safent	Los pods no salen a <code>equipo.lab.safentmcp.com</code> aunque el ordenador desde el que se ejecuta sí: la salida desde el clúster. Con proxy, que deje salir a esa dirección.
Flux no descarga de <code>forgejo.lab.safentmcp.com</code>	Flux no llega al registro. Si el clúster sale por un proxy, ponlo en Safent (proxy , punto 1) y repite el comando.

MENSAJE	QUÉ HACER
⚠ su Flux no tiene proxy	El clúster ya tenía Flux y no sale por el proxy: ponle <code>HTTPS_PROXY</code> y <code>NO_PROXY</code> como indica Flux .
«los nodos no confían en su certificado»	Certificado de la empresa , punto 2, en cada nodo (y reinicia k3s o containerd).
«Flux no confía...» o «El agente no confía en el certificado...»	Certificado de la empresa , punto 1: en Safent. Si ya estaba, pide a sistemas el raíz (y el intermedio, si lo hay) y pega los dos.
Falta la configuración (en Safent)	No es un fallo: crea el secreto con el comando de Configuración de la app .

Probarlo con una app de ejemplo

Safent instala una app suya por el mismo camino que las reales.

1. En Safent, en la ficha del entorno: **Prueba de despliegue** › **Hacer la prueba**. En medio minuto se para en «Falta la configuración».
2. En el terminal, ponle su configuración:

```
echo "MENSAJE=Hola desde el clúster" > safent-ejemplos-prueba-despliegue.env
echo "_huella=$(date +%s)" | kubectl -n flux-system create secret generic safent-
ejemplos-prueba-despliegue-config --from-env-file=safent-ejemplos-prueba-
despliegue.env --from-env-file=/dev/stdin --dry-run=client -o yaml | kubectl
label --local -f - reconcile.fluxcd.io/watch=Enabled -o yaml | kubectl apply -f -
```

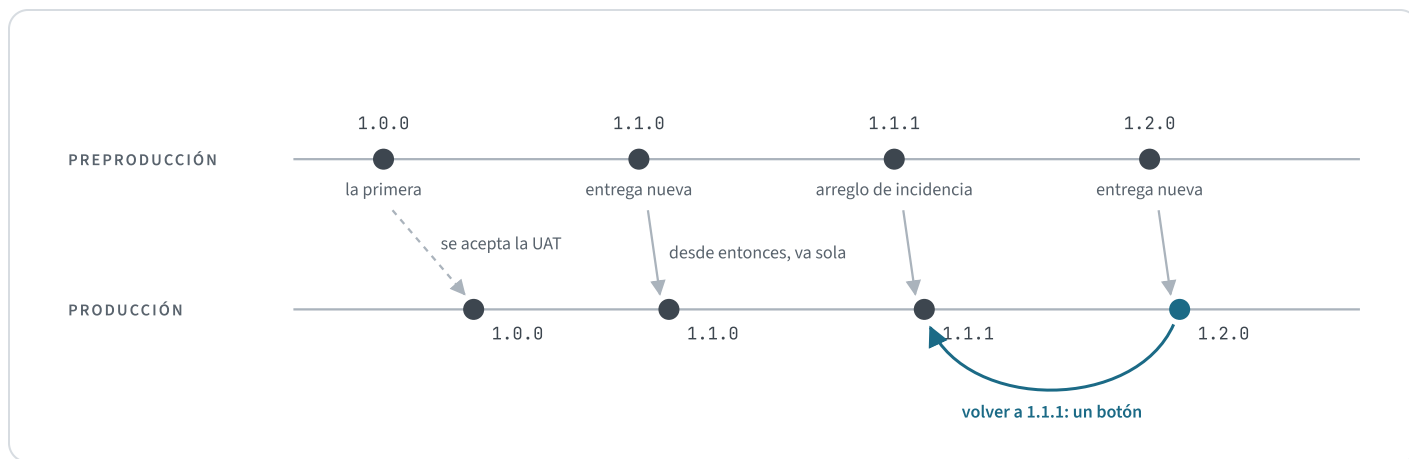
3. En unos 30 segundos sale **Instalada**. Para verla, en el terminal (Ctrl+C para cerrar) y después `http://localhost:8080` en el navegador: dice «Despliegue correcto» con tu mensaje.

```
kubectl -n safent-ejemplos-prueba-despliegue port-forward svc/app 8080:8080
```

4. Para quitarla: **Quitar la prueba** en Safent y, en el terminal, `kubectl -n flux-system delete secret safent-ejemplos-prueba-despliegue-config`.

Versiones, volver atrás e incidencias

Cada versión firmada se guarda; volver a una anterior es un botón.

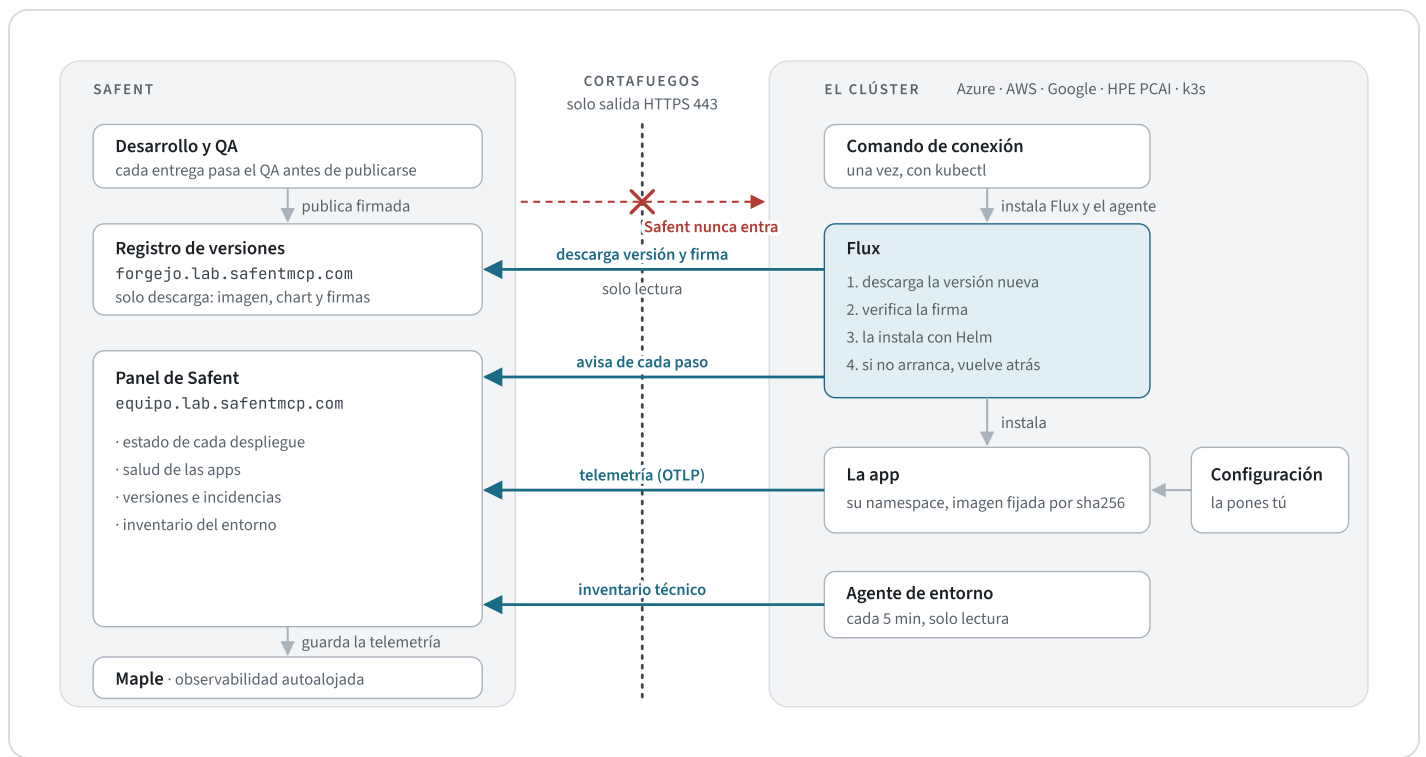


Todas las versiones siguen firmadas y guardadas. Volver a una anterior no toca el entorno: es la misma recogida de siempre, con otro número.

- **Volver a una anterior:** en el Despliegue del repositorio, **Versiones** • **volver a una anterior** > «Instalar en producción» (o en preproducción) junto a la que quieras. Un minuto, sin tocar el clúster.
- **Si una versión no arranca,** Flux vuelve sola a la anterior y Safent la marca como fallida, con el motivo.
- **Una incidencia** se reporta en Safent (menú **Reportar incidencia**). Su arreglo sale como versión de parche (1.1.0 → 1.1.1) y llega sola, como cualquier otra.

Qué ve Safent y qué se queda en el clúster

Nadie entra; todo va firmado; los datos no salen.



Cada flecha azul es una conexión que abre **el clúster** hacia Safent, por HTTPS (443). En sentido contrario no hay ninguna: sin SSH, sin VPN y sin puertos abiertos de entrada.

- **Nadie entra:** ni accesos, ni SSH, ni VPN. Todas las conexiones las abre el clúster, hacia dos direcciones de Safent.
- **Todo firmado:** cada imagen y cada chart se firman al publicarse; el clúster comprueba la firma y, si no cuadra, no instala.
- **La configuración se queda en el clúster**, en un Secret. Safent solo sabe cómo se llaman las variables.
- **Safent ve** el estado de cada despliegue, el inventario técnico (versión, nodos, GPU, almacenamiento; nunca datos) y la salud de las apps (peticiones, errores y tiempos, en la pestaña **Salud** de cada proyecto). La telemetría se guarda en la infraestructura de Safent, no en terceros.
- **En el clúster quedan** Flux y la credencial de solo lectura en `flux-system`, el agente de entorno en `safent-entorno` y un namespace por app.

Sin Internet (planta aislada)

Cada versión llega en un paquete firmado que se instala a mano.

1. Da de alta el entorno con **Conectividad: Aislado**. Safent da un fichero de conexión: en el servidor, `sudo sh conectar-aislado.sh` (Docker) o `kubectl apply -f conectar-aislado.yaml` (Kubernetes).
2. Cada versión se descarga en **Versiones > Paquete offline**. En el servidor:

```
tar xzf paquete-*.tar.gz && cd paquete-*/ && cp app.env.example app.env
```

Pon los valores en `app.env` (si la app pide configuración) y ejecuta `./instalar-offline.sh` : comprueba las firmas e instala.

3. El estado lo deja el agente en un fichero firmado; súbelo en Safent con **Subir estado**.

Pausar las actualizaciones o dar de baja el entorno

Pausar es un comando; dar de baja, un botón y unos comandos de limpieza.

Pausar (lo instalado sigue funcionando; para reanudar, lo mismo con `false`). El nombre es `safent-<cliente>-<entorno>` :

```
kubectl -n flux-system patch kustomization safent-<cliente>-<entorno> --type merge -p '{"spec":{"suspend":true}}'
```

Dar de baja: en Safent, ficha del entorno › **Dar de baja**. Safent deja de entregarle versiones, anula su token y enseña los comandos de limpieza, ya con el nombre del entorno. Ejecutados en el terminal, quitan todo lo de Safent, apps incluidas. Quedan Flux y la configuración de cada app.

Probado de principio a fin en un servidor Ubuntu con k3s, también detrás de un proxy que inspecciona el HTTPS: conectar tarda menos de un minuto y una app queda instalada medio minuto después de poner su configuración. Azure, AWS, Google Cloud y HPE PCAI siguen la documentación oficial de cada proveedor. Safent Developer Team · octubre de 2026.