Manual de usuario

Manual completo de operación del producto: enrolamiento, consola, recetas, respaldos, recuperación, licenciamiento, seguridad, firma de informes e integraciones.

Manual de usuario — SEGRD Abzu v2

Versión del documento: 2026-10-18 · Producto: Abzu v2 (consola + agentes + servicio de licencias, rama `main` integrada) · Idioma: español · Marca: SEGRD

> **Cómo leer este manual.** Describe lo que el producto hace de verdad según su código y su diseño vigente. Cuando una capacidad tiene un límite real — por ejemplo, un ejecutor solo detecta pero no ejecuta, o una prueba contra hardware físico sigue pendiente — el manual lo dice en el mismo apartado, con la etiqueta **Límite verificado**. Nada de lo que sigue promete una función que no esté en el código. Si una pantalla o un mensaje de su instalación difiere, prevalece lo que muestra su instalación y lo registrado en su bitácora de auditoría.

Índice

1. Introducción y formas del producto
2. Requisitos, instalación y enrolamiento de agentes
3. La consola: navegación y paneles
4. Agentes e inventario unificado
5. Recetas: crear, versionar, reconciliar y capturar
6. Catálogo de recetas
7. Respaldos
8. Cinta LTO
9. Recuperación ante desastres (DRP)
10. Licenciamiento y función de pago de VMs
11. Seguridad, roles y cumplimiento
12. Firma de informes y validación «Ver pruebas»
13. Integraciones y notificaciones
14. API pública, SDKs y Terraform
15. Federación de consolas multi-sitio
16. Observabilidad: anomalías, predicciones y KPIs
17. App móvil (PWA)
18. Solución de problemas y mensajes de error
19. Buenas prácticas de operación

20. Glosario

1. Introducción y formas del producto

1.1 Qué es

SEGRD Abzu es una plataforma de continuidad operativa: una **consola central** que gobierna una flota de servidores a través de **agentes ligeros** instalados en cada servidor o clúster. Sus piezas son:

- **Recetas versionadas** (`abzu.recipe/v1`): el estado deseado de cada aplicación — servicios, imágenes, volúmenes, redes, políticas — declarado en YAML. La receta es la fuente de verdad; los artefactos que se aplican (Compose en Docker, manifiestos en Kubernetes) se generan desde ella y no se editan a mano.
- **Reconciliación continua anti-drift**: cada agente compara el estado real contra el deseado y converge o alerta según la política de la receta.
- **Respaldos verificados**: respaldo de bases de datos multi-motor y de volúmenes, con cifrado en flujo, retención y verificación por restauración real en un entorno aislado (sandbox).
- **Recuperación ante desastres (DRP)**: runbooks versionados que reconstruyen en orden — red → volúmenes → datos → aplicaciones — con aprobación humana en la línea base y, en v2, autonomía gobernada por políticas.
- **Licenciamiento por agente**: cada agente enrolado consume un cupo (seat); los contenedores por agente son ilimitados.

El modelo de acceso es deliberadamente distinto al de los paneles que guardan credenciales de administración de todos los servidores en un solo lugar: **no hay llaves SSH centralizadas y no se abre ningún puerto entrante** en los servidores gestionados. El agente siempre inicia la conexión hacia la consola (phone-home) y se autentica con un certificado propio (mTLS). Si un agente se compromete, el alcance es ese servidor, no la flota.

1.2 Las dos formas del producto

El núcleo es el mismo (agente, motor de recetas, runbooks, respaldos, i18n español/inglés). Cambian el empaquetado, la identidad y el licenciamiento:

Aspecto	Forma comercial	Forma gobierno (llave en mano)
Dónde vive	Módulo integrable a la plataforma SEGRD	Instalación autocontenida en la red de la institución, sin depender de ninguna nube
Identidad	La plataforma SEGRD es el proveedor de identidad (OIDC / login nativo servidor a servidor). No se crean usuarios locales duplicados	Usuarios y roles locales + SSO con Entra ID (Microsoft 365) y Google Workspace + LDAP/AD opcional
Licencia	Suscripción anual por agente; validación en línea contra el punto técnico de licencias	

Aspecto	Forma comercial	Forma gobierno (llave en mano)
		Perpetua por agente + soporte anual opcional; archivo firmado que se valida **sin red**
Asignación de licencias	La gobierna el panel de licencias y plataformas del proveedor (ver §10). Abzu no mantiene un registro de asignación paralelo	La misma regla: la asignación manda; la licencia offline firmada la materializa en la instalación
Actualizaciones	Canal firmado en línea	Bundle firmado importable sin conexión
Telemetría	Agregada, con opción por tenant	Ninguna sale de la instalación salvo autorización explícita de la institución
Recetas base	Catálogo de recetas (ver §6)	Biblioteca incluida + bundle offline firmado del catálogo
Notificaciones	Correo y webhooks de eventos hacia la plataforma, más las integraciones de §13	Correo vía Exchange Online y alertas a Teams, más las integraciones de §13 que la institución configure
Cumplimiento	Marco según mercado (ver §11.7)	En República Dominicana: NORTIC. Fuera de RD: nunca NORTIC (ver §11.7)

En todo el manual, cuando un procedimiento difiere por forma, verá dos subapartados: **Comercial** y **Gobierno**. Cuando no los vea, el procedimiento es idéntico.

1.3 Principios que el usuario va a notar

1. **Fail-closed (ante la duda, no actúa).** Si una firma no verifica, una licencia no se puede validar fuera de su gracia, una política es ambigua o un canal cae, el sistema **no ejecuta** la acción y escala. Esto se explica en lenguaje de operación en cada módulo: qué se detiene, qué sigue corriendo y qué debe hacer usted.
2. **Lo que ya corre nunca se detiene por una decisión de control.** Vencimiento de licencia, revocación de un agente o bloqueo de una función de pago detienen lo nuevo (enrolamientos, despliegues, acciones de pago); jamás apagan workloads en ejecución ni tocan datos del cliente.
3. **White-label absoluto.** La consola, el catálogo, la app móvil y este manual usan la marca SEGRD y la marca que cada instalación configure. Los nombres de terceros solo aparecen como **configuración suya** (por ejemplo, el canal de notificaciones que usted conecta), nunca como voz del producto.
4. **Sin secretos visibles.** Ninguna pantalla muestra valores de secretos, contraseñas ni llaves. Las recetas solo contienen referencias (`${secrets.<nombre>}`); la interfaz muestra `••••••` incluso a los administradores.
5. **Todo queda registrado.** Enrolamientos, despliegues, convergencias, aprobaciones, decisiones de DRP, accesos y exportaciones se escriben en el registro de auditoría con cadena de hash (cada

entrada enlaza con la anterior; alterar o borrar una entrada intermedia rompe la cadena y se detecta).

1.4 Roles de quien usa este manual

Rol en la consola	Qué hace normalmente
Operador	Opera el día a día: ve agentes y recetas, despliega, respalda, restaura con las aprobaciones que correspondan
Aprobador	Aprueba ejecuciones de DRP, políticas y sesiones de depuración; puede además operar según su asignación
Auditor	Solo lectura con acceso a auditoría, recertificaciones y exportación de evidencia
Solo lectura	Consulta paneles e inventario; no ejecuta acciones
Administrador de la instalación	Además de lo anterior: configuraciones, firma de informes, integraciones, recertificaciones y validación

La matriz exacta de permisos y la denegación por defecto están en §11.1. Existe además una cuenta de soporte del proveedor con acceso total registrado (ver §11.3): no es un rol asignable desde su consola.

---

2. Requisitos, instalación y enrolamiento de agentes

2.1 Requisitos del agente

- **Sistema:** Linux x86_64 o ARM64. El agente es un binario estático único en Go (sin intérprete ni dependencias de runtime en el servidor destino).
- **Compilación (solo si compila usted mismo):** Go 1.24 o superior. El binario resultante no necesita nada más.
- **Red:** solo salida HTTPS hacia la consola. No se requiere **ningún** puerto entrante. Si su red pasa por un proxy corporativo restrictivo que no admite el canal persistente, el agente usa un mecanismo de reserva (long-poll HTTPS) con el mismo modelo de mensajes y menor rendimiento.
- **Para gestionar contenedores Docker:** acceso al socket del motor (`/var/run/docker.sock`). Sin socket, el agente se degrada con honestidad: reporta inventario vacío de contenedores y sigue funcionando.
- **Para Kubernetes:** el agente se despliega como Deployment con una cuenta de servicio dedicada y permisos mínimos acotados a los espacios de nombres gestionados — nunca administrador del clúster.
- **Para hipervisores, cinta y respaldos:** requisitos adicionales por función en §4.4, §7 y §8.

2.2 Qué es enrolar

Enrolar es dar de alta un servidor: la consola registra el agente, le emite un certificado mTLS propio (validez de 30 días, renovación automática al 70 % de su vida, sin intervención) y el agente abre su canal permanente. Desde ese momento la consola cuenta un seat por ese agente.

Hay dos credenciales de alta:

Credencial	Uso recomendado	Comportamiento
Token de enrolamiento de un solo uso	Alta individual y altas sensibles. **Es la opción recomendada**	Un solo uso, vigencia de **15 minutos**, atado a un tenant y a una etiqueta (p. ej. `produccion`). Al usarse queda **quemado**: no sirve una segunda vez
Llave de flota (llave de onboarding reutilizable)	Altas masivas de una flota	Admite N usos, con etiquetas y fecha de caducidad; es **revocable con un clic** desde la consola

Además, el enrolamiento incorpora el **fingerprint (huella) del servidor**: huella de máquina y, si el agente corre en una nube pública, la **identidad de la nube** detectada desde su servicio de metadatos (proveedor, ID de instancia y región/zona). En un servidor local sin metadatos de nube alcanzables, el resultado «sin nube» es válido y no es un error. La huella ata el seat a esa máquina: clonar un agente no duplica seats — la consola detecta la huella duplicada y la marca para revisión.

2.3 Procedimiento: enrolar un servidor con token de un solo uso

1. En la consola, vaya a **Agentes** y elija **Generar token de enrolamiento**. Indique el tenant (si su cuenta ve más de uno) y la etiqueta. La consola muestra el token **una sola vez**: cópielo en ese momento.
2. En el servidor destino, instale el binario del agente y copie la configuración de ejemplo a `/etc/abzu/agent.yaml`:

```
sudo mkdir -p /etc/abzu /var/lib/abzu-agent
sudo cp agent.yaml.example /etc/abzu/agent.yaml
sudo chmod 600 /etc/abzu/agent.yaml
```

1. Edite `/etc/abzu/agent.yaml` y complete como mínimo:
    1. `console_url`: URL base de su consola, sin barra final.
    2. `agent_name`: nombre legible del servidor (vacío = se usa el nombre del equipo).
    3. `token`: el token del paso 1. **Este archivo debe quedar con permiso 0600** y nunca debe versionarse con valores reales.
    4. `data_dir`: por defecto `/var/lib/abzu-agent`.
    5. `insecure: false`. El valor `true` desactiva la verificación TLS y existe **solo para pruebas locales** contra consolas sin certificado válido. Está prohibido en producción y el propio agente lo marca así en código, registros y documentación.

6. Arranque el agente:

```
# Directo (prueba):
./abzu-agent --config /etc/abzu/agent.yaml

# Como servicio:
sudo cp abzu-agent /usr/local/bin/abzu-agent
sudo cp abzu-agent.service /etc/systemd/system/abzu-agent.service
sudo systemctl daemon-reload
sudo systemctl enable --now abzu-agent
sudo journalctl -u abzu-agent -f
```

1. Qué ocurre por dentro (lo que debe ver):
    1. El agente **genera su par de llaves en el propio servidor**. La llave privada nunca sale de él.
    2. Llama a `POST {console_url}/v1/enroll` presentando el token, la huella y su versión, fijando la huella del certificado de la consola (pinning): si ese certificado cambia sin una rotación anunciada, el agente se negará a conectar.
    3. La consola valida que el token esté vigente y sin usar, registra el agente y devuelve su certificado cliente mTLS. El token queda quemado.
    4. El agente abre su canal con ese certificado. A partir de ahí, **toda** la comunicación es mTLS.
5. Verifique en la consola que el agente aparece en **Agentes** con estado activo y primer latido (heartbeat) recibido, y en el servidor con:

```
abzu-agent status --json --config /etc/abzu/agent.yaml
```

La salida muestra datos **reales** leídos en ese momento (contenedores del socket, CPU/memoria de `/proc`): `agent_id` persistido al enrolar, nombre del equipo, versión, salud y contenedores. En modo demo local (ver §2.5) `agent_id` viene vacío.

Si algo falla: token vencido (pasaron más de 15 minutos) o ya usado → genere uno nuevo; no reintente con el mismo. Error de certificado con `insecure: false` → revise que `console_url` apunte a la consola correcta y que su certificado sea válido; no «resuelva» el problema poniendo `insecure: true` en producción. Detalle completo en §18.

2.4 Procedimiento: altas masivas con llave de flota

1. En **Agentes**, cree una **llave de flota**: defina etiquetas, número de usos o caducidad según su procedimiento interno.
2. Distribuya el instalador con la llave por su canal habitual de despliegue. Cada servidor sigue los pasos 2–5 de §2.3, sustituyendo el token por la llave.
3. Al terminar la ventana de altas, **revoque la llave** desde la consola. Los agentes ya enrolados no se ven afectados: su identidad es su certificado, no la llave.

2.5 Modo demo local y modo sin conexión

- **Modo demo local:** si la configuración no trae token, la consola no responde o el endpoint de enrolamiento no existe, el agente **no se cae**: lo dice con claridad en su registro y continúa escribiendo cada latido en el registro local en vez de enviarlo. Sirve para probar el inventario sin consola. No es un modo de producción.
- **Sin conexión con la consola (modo offline operativo):** el agente sigue aplicando el **último estado deseado conocido** (guardado localmente), encola trabajos y eventos en almacenamiento local persistente y los entrega **en orden** al reconectar. La licencia entra en el periodo de gracia que corresponda (ver §10) con cuenta regresiva visible. Al agotarse la gracia se bloquean los despliegues nuevos; lo que ya corre **jamás se detiene**.

2.6 Revocar un agente

1. En **Agentes**, abra la ficha del agente y elija **Revocar**.
2. Efecto: su certificado entra en la lista de revocación que viaja en el latido (menos de 60 segundos), su canal se cierra y sus trabajos pendientes se congelan. El agente revocado **no puede re-enrolarse solo**: necesita un token nuevo.
3. Lo que **no** hace la revocación: no detiene los contenedores ni las máquinas que ya corrían en ese servidor. Cortar el control no es cortar los datos; retirar los workloads es una decisión operativa separada y explícita.

2.7 Actualización del agente

La consola anuncia las versiones disponibles por el propio canal del agente. El agente descarga la actualización por ese canal, **verifica su firma** y la suma del manifiesto antes de aplicarla, la aplica con sustitución atómica y, si la comprobación de salud posterior falla, **revierte automáticamente** a la versión anterior. La consola puede fijar (pin) una versión por agente o por tenant. Una actualización cuya firma no verifica no se aplica: fail-closed.

---

3. La consola: navegación y paneles

3.1 Acceso e inicio de sesión

Comercial. La pantalla de entrada es el **login nativo de la plataforma**: usted escribe su correo y su contraseña **en la propia consola**, sin ser redirigido a otra pantalla de login. Por detrás, el servidor de la consola valida esas credenciales contra la plataforma en una llamada servidor a servidor. Según su cuenta, la misma pantalla puede pedirle un paso adicional sin sacarle de ella:

Paso adicional	Cuándo aparece	Qué hacer
Código TOTP	Su cuenta tiene segundo factor por autenticador	Escriba el código y continúe
Código por correo		Escriba el código que recibió por correo

Paso adicional	Cuándo aparece	Qué hacer
	Se detecta un contexto nuevo (IP/dispositivo)	
Selección de cuenta	Su correo tiene varias cuentas con acceso	Elija la cuenta en el selector que muestra la propia consola
Cambio de contraseña requerido	La plataforma exige renovarla primero	Siga la indicación que muestra la pantalla; hasta completarlo no se crea la sesión

Gobierno. Login con usuarios locales de la instalación y, si su institución lo configuró, botones de acceso con su proveedor institucional (Microsoft 365 / Google) o directorio LDAP/AD. Las cuentas locales de administrador están sujetas a las reglas de su instalación (ver la nota honesta de §11.6).

Passkey. Si registró una llave de acceso (huella, rostro o llave de seguridad), puede entrar con ella desde la misma pantalla. Registre la primera desde el módulo de seguridad de su perfil, ya dentro de la consola. La ceremonia requiere un contexto seguro (HTTPS); la detección de clonación está activa: si una llave se duplica, el sistema lo detecta y la bloquea.

Sesión. La sesión tiene duración máxima y cierre por inactividad según la configuración de su instalación (en la instalación de referencia: 4 horas en total y 60 minutos de inactividad). El botón **Salir** invalida la sesión en el servidor, no solo en el navegador. Tras varios intentos fallidos seguidos, el inicio de sesión se bloquea temporalmente y responde con limitación de velocidad (error 429, ver §18): espere el tiempo indicado, no insista.

Selección de tenant. Su cuenta puede tener alcance sobre uno o varios tenants (organizaciones/instituciones). Todo recurso — agentes, recetas, secretos, respaldos, políticas — pertenece a un tenant y **no cruza** a otro: el aislamiento se aplica en las consultas, no solo ocultando botones. Un identificador de tenant inexistente muestra un error 404 honesto; un recurso de otro tenant también se responde como no encontrado, para no revelar su existencia (este mismo criterio usa la API pública, §14).

3.2 Mapa de la consola

Zona	Para qué sirve	Apartado
Panel (dashboard)	Estado general en tiempo real: flota, contenedores, respaldos, actividad reciente	§3.3
Agentes	Registro de agentes: estado, versión, último latido, revisión aplicada; enrolamiento y revocación	§2, §4
Inventario	Vista unificada de nodos: contenedores, pods, VMs y contenedores de hipervisor	§4
Recetas	Estado deseado versionado: crear, editar, diff, rollback, aplicar	§5
Catálogo	Recetas publicables por instituciones: publicar, revisar, instalar, bundles offline	§6
Respaldos	Políticas, puntos de restauración, verificación, restauración, informes	§7, §8

Zona	Para qué sirve	Apartado
DRP	Runbooks, políticas de autonomía v2, evaluaciones, vetos, drills	§9
Licencias / Funciones	Estado de la licencia, cupo, funciones de pago y su estado	§10
Federación	Sitios federados, destinos entre sitios y planes DR entre datacenters (si su instalación es central)	§15
Validación	Corridas de validación con su informe «Ver pruebas»	§12.3
Evidencia / Cumplimiento	Exportación de evidencia etiquetada por marco	§11.7
Recertificación	Campañas de revisión de accesos	§11.5
Integraciones	Canales de notificación, suscripciones de webhooks y entregas	§13
Observabilidad	Anomalías, predicciones, recomendaciones y KPIs/KRIs	§16
Auditoría	Bitácora de eventos con cadena de hash, exportable	§11.7
Configuraciones	Ajustes de la instalación, incluida **Firma de informes**	§12.1
Seguridad (perfil)	Sus llaves de acceso (passkeys) y su contexto de seguridad	§3.1, §11
Móvil	La misma cuenta en la PWA: estado, alertas y aprobaciones	§17

No todas las zonas aparecen en todas las instalaciones: el RBAC (§11.1) oculta y, sobre todo, **deniega** lo que su rol no contempla. Que un botón no se vea no es la protección; la protección es que el servidor responde 403 aunque se invoque la ruta directamente.

3.3 Paneles en tiempo real

El panel general y las vistas de detalle se alimentan del latido de los agentes (cada 30 segundos) y de los eventos de la consola:

- **Topología y contadores:** servidores → contenedores/pods/VMs, con aparición escalonada de tarjetas y contadores animados. Los datos se refrescan solos (en la instalación de referencia, el panel general cada ~20 segundos).
- **Línea de tiempo:** despliegues, convergencias, respaldos, restauraciones y drills como eventos con su progreso y su resultado.
- **Mapa de calor de RPO:** por servicio, cuánto tiempo de datos se perdería hoy si fallara — el insumo visual de las decisiones de respaldo.
- **Modo presentación:** un interruptor puebla la consola con datos de demostración animados **sin tocar datos reales**, para presentaciones. Desactívelo para volver a los datos reales; ninguna acción del modo presentación opera sobre la flota.

> **Rendimiento y accesibilidad:** las animaciones usan solo transformaciones y opacidad, las listas largas se virtualizan y la consola respeta la preferencia del sistema de reducir el movimiento (`prefers-reduced-motion`). La consola es operable por teclado; el objetivo de accesibilidad de la forma gobierno es el nivel AA (ver §11.7).

Si un panel muestra «sin datos», significa exactamente eso: la fuente no tiene registros. La consola **no inventa cifras**: un KPI sin fuente, un agregado de catálogo sin mediciones o un respaldo sin manifiesto se presentan como ausencia explícita, nunca como un cero maquillado.

---

4. Agentes e inventario unificado

4.1 El latido (heartbeat)

Cada 30 segundos (configurable entre 15 y 60) cada agente reporta: su identidad y versión, la revisión de receta aplicada por workload, un resumen del inventario (contenedores/servicios, imágenes, volúmenes, espacios de nombres — resúmenes con hashes, no volcados completos), la salud del nodo (CPU, memoria, disco, alcance al motor de contenedores, profundidad de su cola local) y, como respuesta, recibe el estado de su licencia. Ese latido es también el canal por el que viajan la lista de revocación y los anuncios de actualización.

Estados de un agente en la consola: **activo**, **degradado** (responde, pero algo falla — p. ej. sin alcance al motor), **sin conexión** (no reporta; sus workloads siguen corriendo, ver §2.5) y **revocado**.

4.2 Inventario unificado

La vista de inventario reúne en una sola tabla de nodos, con la misma experiencia de operación:

- Contenedores **Docker** reportados por el agente (nombre, imagen, estado).
- **Pods** y workloads de **Kubernetes** de los clústeres gestionados.
- **Máquinas virtuales (VM) y contenedores de hipervisor (LXC)** de los hipervisores conectados (§4.4), etiquetados con su hipervisor y su host.

Cada nodo muestra su origen (qué agente o qué hipervisor lo reporta). Lo que una fuente no puede dar, se muestra ausente — por ejemplo, las direcciones IP de una VM cuyo agente invitado no está instalado — en vez de estimarse.

4.3 Operaciones sobre contenedores y pods

Desde la ficha del nodo, el operador puede ejecutar las operaciones tipadas que la receta y su rol permiten (aplicar la revisión deseada, iniciar/detener/reiniciar donde la superficie lo habilita). El agente **no ejecuta una shell arbitraria**: solo operaciones derivadas de recetas y trabajos tipados, contra una lista permitida. Toda operación queda en auditoría con actor, objetivo y resultado.

4.4 Hipervisores y máquinas virtuales — función de pago

La gestión de VMs/LXC es una **función de pago** (su estado de licencia se explica en §10.4; en prueba funciona todo, con modal de conversión que no bloquea). Con ella activa, la instalación descubre y opera VMs/LXC de los hipervisores siguientes. **Alcance vigente:** la gestión disponible hoy es la de **Proxmox VE**; **VMware, Hyper-V, Xen y KVM/libvirt genérico** forman parte del programa de hipervisores comprometido (hoja de ruta), y las fichas de esta tabla describen el diseño de cada integración:

Hipervisor	Cómo se conecta	Qué hace	Límites verificados
Proxmox VE	API REST del clúster (token de API o ticket de usuario)	Inventario de VMs QEMU **y contenedores LXC**, estado, encender/apagar	La acción de apagar es **apagado duro**; el apagado ordenado del sistema invitado es una acción separada. Las IP de una VM QEMU dependen de que tenga agente invitado instalado. **A diferencia de otros productos de respaldo, aquí los LXC sí entran en el inventario y en el respaldo (§7.8)**
VMware	API de **vCenter** (ESXi es cerrado: no admite agente en el host)	Inventario básico de VMs y encender/apagar por API	Sin IPs en la lista básica (la API de listado no las trae). Las VMs también pueden llevar agente dentro de cada VM
Hyper-V	Agente/API en el host Windows (PowerShell)	Inventario y encender/apagar	**Solo en hosts Windows** con el módulo de Hyper-V. En cualquier otro sistema el proveedor se reporta como no disponible con su motivo — error honesto, nunca datos inventados. El identificador del nodo es el nombre de la VM
KVM / libvirt (genérico)	Herramienta `virsh` en el host	Inventario (`list` / `dominfo` / `domifaddr`) y encender/apagar	Si `virsh` no existe en el host, el proveedor se reporta como no soportado

Procedimiento general:

1. En la ficha de hipervisores, registre la conexión (dirección del API y referencia a la credencial). **Las credenciales se guardan cifradas (AES-256-GCM)** en un almacén propio del agente, referenciadas por identificador: nunca en claro en la configuración, en registros ni en pantalla. La llave maestra del almacén llega por entorno o por un archivo con permiso 0600; un archivo con permisos más abiertos se rechaza.
2. El agente descubre los nodos y los presenta en el inventario unificado con su tipo (`vm` / `lxc`), host, hipervisor, estado, CPU, memoria e IPs cuando la fuente las da.
3. Las acciones de energía (encender/apagar) pasan por el **gate de entitlement**: con la función bloqueada, el inventario se sigue viendo pero las acciones se deniegan con su motivo (§10.4). Donde una operación no sea posible por API en un hipervisor, la consola lo muestra como limitación; no la simula.

> **Límite verificado — Podman:** el agente **detecta** sockets de Podman (con y sin root) y lo reporta con fines de diagnóstico, pero el ejecutor de contenedores usa un socket fijo y **no ejecuta sobre Podman** en esta versión. No presente Podman como motor soportado: es detección informativa.

4.5 Agentes en la nube

El mismo binario corre en AWS, Azure y GCP sin abrir puertos entrantes (la salida HTTPS existe en toda nube). En el enrolamiento, la detección de metadatos de la nube (§2.2) enriquece la huella con

proveedor, instancia y región. Donde la nube lo aporta, los snapshots de volúmenes de la nube pueden usarse como destino de respaldo adicional y los secretos pueden **referenciarse** desde el almacén de secretos de la nube sin copiar sus valores a la consola. Las etiquetas de costo por workload, cuando la nube las expone, se asocian al workload correspondiente.

> **Límite verificado:** la detección de metadatos usa intentos cortos (≤ 1 s por proveedor, en orden AWS → Azure → GCP). En redes con proxy obligatorio hacia las direcciones de metadatos, la detección puede no ver la nube y el agente se tratará como local. Ese resultado vacío es seguro: no bloquea el enrolamiento.

---

5. Recetas: crear, versionar, reconciliar y capturar

5.1 El formato en una página

Una receta es un YAML `abzu.recipe/v1` que declara el estado deseado:

```
 apiVersion: abzu.recipe/v1
kind: Recipe            # Recipe | Runbook
metadata:
  name: tienda-web      # minúsculas, números y guiones; máx. 63
  version: 1.4.0        # versionado semántico
  labels: { entorno: produccion }
spec:
  targets: [docker, kubernetes]   # dónde puede desplegarse
  driftPolicy: auto               # auto | manual | report
  secrets:
    - name: db-password           # solo la REFERENCIA; el valor vive cifrado
  services:
    web:
      image: registro.interno/tienda-web:1.4.0   # etiqueta explícita, nunca
implícita
      replicas: 2
      ports: ["8080:80"]
      env:
        DATABASE_URL: "postgres://tienda:${secrets.db-password}@db:5432/tienda"
      depends_on: [db, cache]
      resources: { cpu: "500m", memory: "512Mi" }
      healthcheck: { http: "/salud", interval: 30s }
    db:
      image: postgres:16
      stateful: true              # con datos: convergencia automática prohibida
      driftPolicy: manual
      volumes:
        - name: datos-pg
```

```
        path: /var/lib/postgresql/data
        backup: { schedule: "0 2 * * *", retentionDays: 7 }
networks: [tienda-net]
```

Reglas que el sistema hace cumplir (no son consejos):

- **Ningún secreto en claro.** Solo referencias `${secrets.<nombre>}`, declaradas en `spec.secrets`. Una receta con un valor en claro se rechaza — también en el catálogo (§6).
- **Imágenes con etiqueta explícita** (o digest `imagen@sha256:…`). Nada de «la última» implícita.
- **Compilación determinista:** la misma receta y versión producen artefactos byte a byte idénticos.
- La especificación pública completa del formato vive en la documentación del producto (`docs/spec/abzu.recipe-v1.md` del repositorio) y es la referencia normativa para autores.

5.2 Crear y versionar una receta

1. En **Recetas → Nueva**, escriba el YAML o parta de una plantilla de un clic. El editor valida con el analizador estricto del motor: los errores se muestran **en español, con el campo concreto**, antes de guardar.
2. Al guardar, la receta nace con su primera **revisión**. El estado deseado es git-backed: **cada cambio es un commit** con autor, fecha y mensaje.
3. Para cambiarla, edite y guarde: se crea una **revisión nueva e inmutable**; la anterior no se reescribe. La consola muestra el **diff visual** entre revisiones.
4. **Rollback en un clic:** volver a una revisión anterior crea un commit nuevo que la reaplica — nunca reescribe la historia. Lo que se deshace queda tan auditado como lo que se hizo.

5.3 Aplicar una receta (despliegue)

1. Asigne la revisión deseada a un servidor o clúster y elija **Aplicar**.
2. Antes de tocar el motor, la **admisión** puede escanear las imágenes de la receta en busca de vulnerabilidades (§11.6): una imagen en cuarentena según la política **bloquea la aplicación sin tocar el motor** y lo registra. Si el escáner no está disponible, la política por defecto es **fail-closed** (no se aplica sin escanear) salvo que su instalación haya configurado otra cosa a sabiendas.
3. El trabajo entra en la **cola persistente por agente**: con orden, reintentos con espera creciente y dead-letter tras agotar los fallos. Cada trabajo lleva un identificador y el agente lo ejecuta **una sola vez** (idempotencia): si no hay acuse, el trabajo se reentrega al reconectar, no se duplica.
4. El agente compila/aplica: en Docker vía la API del motor sobre su socket (nunca por shell); en Kubernetes con aplicación del lado del servidor y sus permisos mínimos. Los registros del despliegue y las métricas llegan en tiempo real a la consola por el canal del agente.

5.4 Reconciliación anti-drift

En ciclo continuo (cada 60 segundos, y al instante por watches en Kubernetes) el agente compara el estado real (hash canónico por workload) contra el deseado y actúa según la política:

Política	Comportamiento	Cuándo se usa
`auto`	Converge solo al estado deseado y lo registra	Por defecto en servicios **sin** estado
`manual`	Alerta en la consola y **espera aprobación**	**Obligatoria en todo lo que tenga datos** (`stateful: true`): el agente jamás toca datos por su cuenta, aunque la receta diga otra cosa
`report`	Solo registra la desviación	Auditorías o congelamientos de cambio

Toda convergencia o desviación queda en auditoría con el antes y el después. Un drift en un workload crítico también puede generar alerta accionable en la app móvil (§17) y en sus integraciones (§13).

5.5 Captura inversa: de la realidad a la receta

Cuando un cambio se hizo **fuera** de la consola (el clásico arreglo de emergencia a las 3 a.m.), la captura inversa evita que quede como drift eterno e invisible:

1. En la ficha del agente o del workload, elija **Capturar estado real** (`recipe capture` ; puede acotar por contenedores o capturar todos).
2. El agente fotografía el estado real — contenedores/manifiestos vivos, variables **no secretas**, volúmenes — y genera una **revisión borrador** de la receta. Los secretos aparecen solo como referencias enmascaradas.
3. Un humano revisa el borrador (diff contra la revisión vigente) y decide **promoverlo** a estado deseado o descartarlo. Sin aprobación, el borrador no cambia nada.

La captura inversa es también el **motor de migración** de instalaciones existentes: convierte workloads que ya corren en recetas versionadas, que es lo que hace confiable al DRP (§9): la receta vuelve a representar la realidad.

5.6 Secretos de las recetas

- En reposo: cifrados por instalación. En tránsito: solo por el canal mTLS y ya cifrados. En uso: el agente los descifra **en memoria** con su llave de identidad y los inyecta; jamás se escriben a disco en claro ni aparecen en registros o pantallas.
- **Rotar un secreto** = crear una nueva revisión de la receta que lo referencia; el agente la aplica con reinicio rodante de los servicios afectados.
- Ni siquiera un administrador puede ver un valor desde la consola: la interfaz y la API lo enmascaran siempre.

---

6. Catálogo de recetas

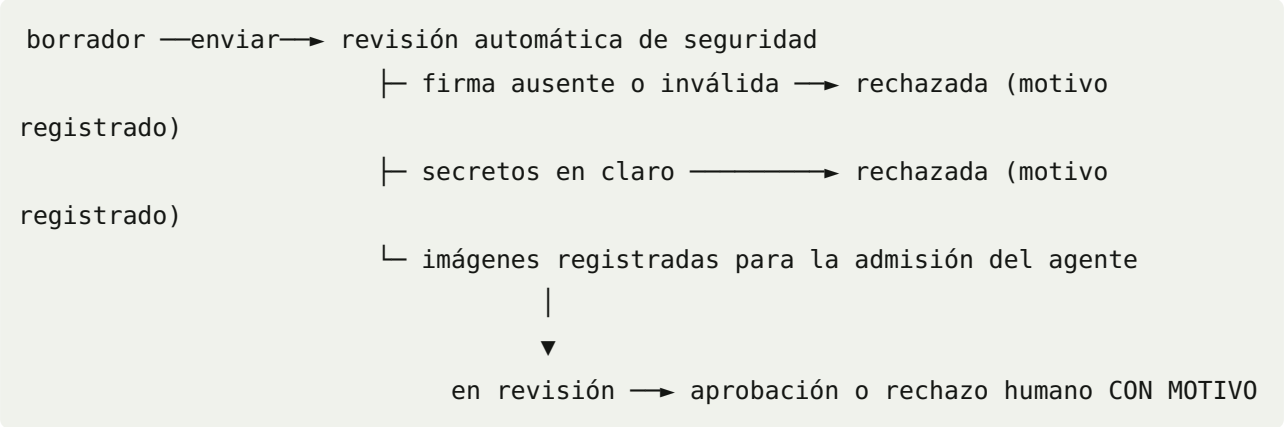
El catálogo es el mecanismo para que una institución **publique** una receta pulida y otras la **instalen** con garantías. No es una tienda de estrellas: sus señales son operativas (§6.4).

6.1 Publicar una receta

1. En **Catálogo → Nueva entrada**, aporte: publicador (su identificador estable), nombre y versión semántica de la publicación, registro de cambios (changelog) en texto claro, el YAML completo de la receta y la **compatibilidad declarada**: destinos (`docker` / `kubernetes`), motores de base de datos que soporta y versión mínima de agente.
2. Cada cambio publica una **entrada nueva e inmutable** con su propia versión; no se edita una entrada ya publicada.
3. La entrada se **firma con la llave del publicador** (firma digital ed25519 sobre un contenido canónico que incluye nombre, versión, la huella del YAML, la del changelog y la compatibilidad). La firma cubre también la compatibilidad: nadie puede alterarla sin invalidarla.

> **El publicador es su llave.** Su identidad pública es la huella (SHA-256) de su llave pública, que debe fijarse por canales independientes para que otras instalaciones la reconozcan. Guarde su llave privada como guardaría cualquier llave de firma: sin ella no podrá publicar nuevas versiones como ese publicador.

6.2 Revisión: de borrador a aprobada



- Todos los estados y motivos quedan **persistidos y auditados**.
- El escaneo de vulnerabilidades de las imágenes ocurre además en el **agente al aplicar** (§5.3): una entrada aprobada con una imagen en cuarentena **no se instala**.
- Solo es instalable una entrada **aprobada + con firma verificada + sin secretos en claro**.

6.3 Instalar desde el catálogo

1. En **Catálogo**, abra la entrada aprobada y elija **Instalar** indicando el destino.
2. La **compatibilidad se aplica, no solo se muestra**: si el destino no está entre los destinos declarados, el motor de BD no coincide o la versión del agente es menor que la mínima declarada, la consola **bloquea la descarga** y responde el motivo, sin entregar el YAML.
3. El agente repite la verificación **antes** de instalar (puerta de instalación, fail-closed), en este orden conceptual: estructura y versionado válidos → **firma del publicador válida contra las llaves confiables configuradas en ese agente** (sin firma, firma inválida, publicador no confiable o **cero**

llaves confiables configuradas = rechazo) → sin secretos en claro → compatibilidad. Solo tras esa puerta la instalación sigue el flujo normal de recetas (§5.3), incluida la admisión de imágenes.

4. Configure en cada agente el directorio de llaves confiables del catálogo (`catalog_trusted_keys_dir` en `agent.yaml`, un archivo de llave pública por publicador). **Vacío significa que ese agente no instala nada del catálogo.** Es deliberado.

6.4 Señales operativas (ratings honestos)

Por entrada, el catálogo puede mostrar agregados **operativos y anonimizados**: instalaciones activas, tasa de drills de restauración exitosos y última verificación. Si una instalación no tiene datos reales de un agregado, muestra **«sin datos»**. Publicar cifras sin medir está prohibido por la especificación del catálogo.

6.5 Bundle offline del catálogo (gobierno)

Para instalaciones sin red:

1. **Exportar:** en **Catálogo → Exportar**, la consola genera un **bundle firmado** (formato `abzu-catalog-bundle/v1`, un archivo transportable) con los publicadores (y sus llaves) y las entradas con su firma individual.
2. Transporte el bundle por su canal aprobado hasta la instalación aislada, junto con — **por un canal separado** — la huella de la llave del emisor del bundle.
3. **Importar:** en **Catálogo → Importar**, la importación es **fail-closed y atómica**: (1) la huella del emisor debe coincidir con la fijada por el canal separado; (2) la firma del bundle debe verificar; (3) la firma de **cada** entrada debe verificar contra su publicador y ninguna puede contener secretos en claro. **Cualquier fallo = no se importa nada.**

---

7. Respaldos

7.1 Modelo general

El módulo de respaldos vive en el **agente**; la consola programa y muestra, y **nunca toca credenciales de bases de datos**. Principios que aplican a todo lo de este capítulo:

- **Todo respaldo viaja en flujo (stream), sin archivos temporales en el disco del host:** herramienta nativa → **cifrado AES-256-GCM en flujo** → subida por partes al destino. La memoria que usa está acotada a una parte más un bloque de cifrado.
- **Manifiesto por respaldo:** junto a cada objeto se guarda su manifiesto JSON con motor y versión, base, tiempos, tamaños (cifrado y en claro), huella SHA-256 del artefacto cifrado, formato, conteo de filas por tabla y, tras verificar, el **tiempo de restauración medido** (su RTO real).
- **La receta declara qué respaldar, nunca con qué clave.** Las credenciales las resuelve el agente dentro del propio contenedor de la base de datos, desde su entorno.

- **Cifrado no desactivable en la forma gobierno:** también para los snapshots de volúmenes (§7.7). Un respaldo marcado como comprometido **se marca y se alerta; nunca se muta ni se borra** (§7.6).

7.2 Motores de base de datos

Motor	Herramienta	Formato	Incrementales / PITR
PostgreSQL	`pg_dump -Fc` (formato personalizado) ejecutado dentro del contenedor de la BD	`pgdump-custom`	Base con copia física + segmentos WAL archivados (ver §7.4)
MySQL / MariaDB	`mysqldump --single-transaction` con rutinas, disparadores y eventos	`mysqldump`	Base con posición de binlog capturada + binlogs cerrados (ver §7.4)
SQLite	Control de consistencia del WAL + volcado (`.dump`)	`sqlite-dump`	Siempre completo (los SQLite reales son pequeños; el incremental no aporta)
SQL Server (MSSQL)	`BACKUP DATABASE` / `BACKUP LOG` vía la herramienta del propio contenedor	`mssql-bak` / `mssql-trn`	Log backups = el incremental nativo; restauración con recuperación (ver §7.4)

La resolución de la herramienta sigue este orden: (1) dentro del contenedor de la BD, si trae el cliente (las imágenes oficiales lo traen); (2) contenedor auxiliar de herramientas; (3) binario del host. La detección confirma la versión del motor y elige las opciones compatibles. En la receta, la base se declara así:

```
databases:
  - name: lbc_db
    engine: postgres
    container: lbc_db
    schedule: "0 2 * * *"
    retention: { daily: 7, weekly: 4, monthly: 6 }
    destination: repositorio-objetos
    verify:
      sandbox: true
      portals: ["https://app.cliente.com/health"]
```

7.3 Crear una política de respaldo

1. En **Respaldos → Políticas**, cree la política: workload/base, motor, destino, programación (expresión cron) y **retención GFS** (abuelo-padre-hijo: diarios, semanales, mensuales y anuales que se conservan).

2. **Punto GFS sellado:** un punto creado en domingo, día 1 o 1 de enero queda clasificado y **sellado** hasta que vence su periodo completo: la retención corta **no lo poda antes**, aunque le tocara por días. Si el destino tiene inmutabilidad (§7.5), el sello se apoya en ella.
3. **Ventana de respaldo:** declare la ventana permitida (p. ej. `22:00-06:00`; admite cruce de medianoche). Fuera de ventana, el trabajo **no arranca**: espera y reporta «esperando ventana». Los servicios que operan 24/7 no se saturan por un respaldo fuera de hora.
4. **«Respaldar ahora»:** desde la ficha del workload, dispare un respaldo inmediato sin esperar a la programación (es la vía manual de forzar el trabajo, equivalente a saltarse la ventana de forma explícita y auditada).
5. **Encadenamiento:** una cadena declarativa puede ordenar los pasos de una misma corrida — respaldo → copia al segundo destino → verificación → chequeo de integridad — con propagación de fallo: si un paso falla, la corrida lo reporta y no finge el éxito de los siguientes.
6. La política es parte del estado deseado: si alguien cambia la programación a mano, la reconciliación lo restaura y lo audita (anti-drift también para los respaldos).

7.4 Incrementales y recuperación a un instante (PITR)

Sobre la misma interfaz de drivers de §7.2, sin cambiarla:

Motor	Base	Incremental	Restaurar a un instante
PostgreSQL	Copia base física con WAL en flujo	Segmentos WAL archivados tras forzar su rotación. Requiere en la BD: archivado activado y nivel de WAL suficiente; si no se cumple, el trabajo lo dice y no promete incrementales	En sandbox: datos de la base + WAL + objetivo por **fecha/hora o por posición (LSN)**; al alcanzarlo, el motor se promueve
MySQL/ MariaDB	Volcado con la posición de binlog capturada del propio flujo	Binlogs cerrados no cubiertos, tras rotarlos	Restaurar la base y reproducir los binlogs hasta la **fecha/hora o posición** pedida
SQL Server	`BACKUP DATABASE` completo (garantiza modelo de recuperación completa) y captura de sus LSN	`BACKUP LOG`	Restaurar la base sin recuperar, aplicar los logs hasta el instante (`STOPAT`) y cerrar con recuperación
SQLite	—	— (siempre completo)	Al último respaldo

Procedimiento PITR desde la consola:

1. En **Respaldos**, abra el workload y su **cadena**: verá la base y los incrementales encadenados (cada eslabón es un manifiesto normal con su padre y el bloque de cadena: tipo, identificador de cadena, secuencia y posición por motor).
2. Elija **Restaurar a un instante** e indique la fecha/hora objetivo (y la posición/LSN si su procedimiento la usa).

3. El sistema restaura **primero en sandbox** (§7.6) y, si usted aporta una **sonda** — una consulta de solo lectura —, la ejecuta en el sandbox y le muestra las filas que existían en ese instante: es la evidencia de que el punto elegido es el correcto, antes de tocar producción.
4. **Retención consciente de cadenas:** la poda GFS nunca deja una cadena rota: si un eslabón se conserva, se conservan su base y todos los incrementales intermedios; una cadena sin descendientes conservados se poda entera.

7.5 Destinos, segunda copia e inmutabilidad (regla 3-2-1)

Un respaldo con un solo destino no cumple ninguna regla seria de continuidad. El módulo trabaja con destinos intercambiables — los trabajos, los manifiestos y la retención no conocen el tipo concreto:

- **Objeto compatible con S3:** el destino habitual.
- **Archivo local / medio offline:** un directorio en un medio desconectable = la copia **offline / air-gap** contra ransomware.
- **Cinta:** ver §8.
- **Sitio federado:** el almacenamiento de otro sitio de la federación (§15).

Segunda copia (backup copy). Configure en la política un segundo destino, inmediato (al crearse el punto) o programado. La copia replica artefacto + manifiesto **en flujo verificando la huella durante la copia**: si el hash no coincide, la copia se aborta y se borra el parcial — nunca queda una «copia» corrupta con aspecto de buena. La copia queda registrada en el manifiesto, con su destino y su retención propia.

Inmutabilidad (Object Lock). Para que ni un atacante con credenciales del agente pueda borrar un punto sellado:

1. El bucket de destino debe **nacer** con Object Lock activado (no se puede activar después en un bucket existente). El servicio lo crea así cuando la política lo pide.
2. Al escribir el punto se le aplica retención: modo **GOVERNANCE por defecto**; **COMPLIANCE configurable** para los puntos críticos. La fecha de retención es el fin del sello GFS del punto, o el periodo que fije la política.
3. **Lea esto antes de elegir modo:** en un bucket con versionado, un borrado simple solo crea una marca de borrado — la versión retenida sigue legible y su huella coincide con el manifiesto — y el **borrado permanente de la versión protegida falla** («el objeto está protegido WORM»). Pero GOVERNANCE **sí admite excepción** para credenciales con un permiso especial de bypass (p. ej. el administrador del almacenamiento). Por eso el patrón obligatorio es el de **repositorio endurecido**:

- Bucket **dedicado por agente/tenant**, creado con Object Lock.
- Credenciales **de un solo uso por agente**, con permiso mínimo solo sobre su bucket (poner/leer/listar) y **sin** permiso de bypass ni de administración. Las credenciales raíz del almacenamiento **nunca** las usa un agente.
- Retención de Object Lock ≥ retención GFS del punto sellado.
- La segunda copia vive en otro sitio/bucket con sus propias credenciales: comprometer un agente no expone la copia.

- Para puntos sellados críticos, use **COMPLIANCE**, que no admite bypass.

Tiering (niveles). Una política de edad puede mover o copiar los puntos antiguos del destino caliente al frío: modo copiar (quedan en ambos) o mover (se borra del caliente **solo tras re-leer el objeto en el frío y verificar su huella**). Los puntos sellados o con Object Lock vigente **se saltan y se reportan**, no se fuerzan.

7.6 Verificación en sandbox y «Ver pruebas»

Un respaldo que nunca se restaura es una promesa, no un respaldo. La verificación de Abzu restaura de verdad:

1. El contenido descifrado se escanea con el antivirus disponible en el host del agente. Si el escáner no está, el informe lo dice (`skipped`), sin fingir que se escaneó. Si detecta malware: el punto se marca **comprometido** (§7.6.1), se alerta y el original **no se borra ni se muta**.
2. Se levanta un **sandbox desechable sin red hacia producción** con la imagen del contenedor de origen.
3. Se espera a que el motor del sandbox responda sano.
4. Se restaura el artefacto y se **cuenta fila a fila por tabla**: la igualdad contra el manifiesto debe ser **exacta**.
5. Se mide el tiempo desde que arranca el sandbox hasta el conteo verificado: ese es el **RTO medido** que queda en el manifiesto.
6. En la verificación extendida se añaden: pruebas HTTP a los portales declarados en la receta (código esperado, contenido y latencia), **captura real de pantalla** del portal y el informe de salud automático que acompaña al respaldo.
7. El sandbox se destruye siempre, pase lo que pase.

SQLite verifica restaurando en un archivo temporal del host (su sandbox natural: sin red ni contenedor) y borrándolo al terminar.

Verificación como política. La verificación se programa **por workload** (frecuencia y alcance: restauración + conteos, HTTP, antivirus), no solo como drill global. Su resultado alimenta el informe «Ver pruebas» de esa corrida (§12.3) y el KPI de restores verificados (§16).

Chequeo de integridad programado. Además de restaurar, un trabajo periódico **re-lee y re-hashea** cada artefacto en flujo y lo contrasta con su manifiesto, sin modificar nada: reporta cada punto como íntegro, corrupto o ausente. Es el complemento barato que detecta la corrupción silenciosa con el tiempo.

Informe periódico de respaldos. Desde los manifiestos, la consola genera por tenant/workload el informe de cobertura: último respaldo y su estado, RPO cumplido contra el objetivo, puntos infectados, copias e integridad del último chequeo, en formato legible (Markdown/JSON). Es la evidencia periódica que la dirección y los auditores piden (ver §11.7).

7.6.1 Puntos infectados y restauración segura

- Si una verificación marca un punto como **infectado**, el estado queda en su manifiesto y **se propaga a todos sus destinos/copias**. La consola lo muestra y el DRP lo respeta.
- Restaurar un punto infectado está **bloqueado por defecto**: exige una aprobación explícita de riesgo (en la operación del agente, la confirmación expresa correspondiente). El sistema registra quién aceptó ese riesgo.
- Antes de restaurar a producción, la **restauración segura** repite el pre-chequeo sobre el punto elegido justo antes de ejecutarla — la verificación pudo correr hace días —: integridad contra el manifiesto y re-escaneo. Si el pre-chequeo falla, no se restaura.

7.7 Volúmenes, archivos del host y catálogo global de puntos

- **Snapshots de volumen:** el contenido de un volumen (o de cualquier directorio) se archiva en flujo, se **cifra siempre** (en gobierno no es desactivable) y se sube con su manifiesto, que incluye el **índice por archivo** (ruta, tamaño y huella de cada archivo en claro).
- **Restaurar un solo archivo:** desde el manifiesto, elija el archivo y **Restaurar archivo**: el sistema recorre el flujo descifrado y extrae **solo ese archivo**, verificando su huella, sin restaurar el volumen entero ni descargar el artefacto a disco. Nota honesta de coste: el recorrido es secuencial, así que un archivo al final de un snapshot enorme tarda proporcionalmente.
- **Respaldo de rutas del host:** declare una lista de rutas (archivos/directorios del host o de un recurso compartido) y se respaldan con el mismo pipeline y el mismo restore por archivo.
- **Catálogo global de puntos:** todos los manifiestos (bases y volúmenes, de todos los destinos) se indexan en un catálogo buscable por workload, fechas, estado (`ok` / `infected` / `sealed`), motor y tipo. Desde el catálogo se llega al punto concreto — y, si vive en cinta, a qué cartucho montar (§8).

7.8 Respaldo de VMs de hipervisor y de la propia consola

- **VMs Proxmox (QEMU y LXC):** con la función de hipervisores activa (§4.4), las VMs y los contenedores LXC de Proxmox se protegen vía su API (snapshot + exportación). Los **LXC incluidos** son un diferenciador real frente a productos que solo respaldan VMs.
- **VMs de otros hipervisores con CBT** (VMware/Hyper-V): clasificado como iteración posterior (v1.1 en la planificación de brechas); mientras tanto, una VM legada sin receta se protege con el **agente dentro de la VM** (sus bases por §7.2 y sus rutas por §7.7). No prometa CBT donde el código no lo trae.
- **La propia consola se respalda a sí misma** con el mismo motor (su base, su configuración y sus recetas) y puede restaurarse desde ese respaldo. Sin esto, el DRP del producto sería teatro.
- **Límite de ancho de banda:** con destino fuera del sitio, el agente puede limitar la velocidad de subida por horario (limitación por token bucket en el subidor), para que un respaldo no sature la producción.

7.9 Respaldo continuo (CDC)

Para workloads cuyo RPO objetivo sea de **minutos o segundos**, el respaldo continuo envía los **segmentos de log** (WAL en PostgreSQL, binlogs en MySQL/MariaDB) a medida que se cierran, en vez de esperar al próximo incremental programado:

1. El servicio de envío descubre los segmentos nuevos desde la última posición confirmada.
2. Sube cada segmento **en orden** al destino (el CDC no conoce el transporte: objeto, archivo o cinta).
3. **Verifica** la subida releyendo el objeto y comparando su huella.
4. **Solo tras verificar, confirma la posición** (persistida atómicamente, segmento a segmento). Un fallo deja la posición en el último segmento verificado: al reintentar se re-sube el segmento en curso, **sin huecos**.
5. La métrica que manda es el **lag**: segundos entre «ahora» y el fin del último segmento confirmado = su **RPO real en segundos**. Si el servicio de envío cae, el RPO crece en silencio: **vigile y alerte el lag**.

El CDC no restaura: garantiza los eslabones. Cada segmento corresponde a un eslabón de la cadena de §7.4, y el PITR es el mismo — el sistema valida que base + segmentos forman una cadena ordenada y restaurable; si la cadena está rota, **lo dice** en vez de prometer un PITR imposible.

Para qué workloads sí: bases transaccionales donde perder 15–60 minutos cuesta más que operar el bucle (pagos, órdenes, inventarios) y con volumen de log moderado — una base grande con poca escritura es el caso ideal. **Para cuáles no:** escritura masiva por lotes (el log crece como los datos y el CDC degenera en copiar todo, continuamente), bases pequeñas o poco críticas con RPO de 1 hora aceptable (bastan los incrementales), y entornos sin nadie que atienda la alerta de lag.

> **Límite verificado — CDC:** quedan fuera de este alcance el streaming de WAL por protocolo de replicación y la captura fila a fila tipo replicación lógica: son otro producto, no respaldo continuo. Y la prueba de extremo a extremo contra motores reales (archivado WAL de un PostgreSQL vivo y rotación de binlogs de un MySQL vivo, con PITR cronometrado) **queda pendiente de un entorno con ambos motores**: las pruebas actuales son unitarias con segmentos sintéticos y destino local de archivos (descubrimiento, orden, confirmación solo tras subida verificada, reanudación sin huecos tras un fallo inyectado y cálculo de lag). No presente el CDC como verificado contra motores vivos hasta que esa prueba exista.

---

8. Cinta LTO

8.1 Qué hay y qué no hay

- **Hoja de ruta comprometida (línea v1.1):** respaldo a cinta **básico con driver genérico**: servidor de cinta Linux, pools simples, GFS de cinta simple, catálogo por código de barras (barcode) y restauración con staging a disco.
- **Hoja de ruta comprometida (v2):** la infraestructura completa construida **sobre** lo anterior: pools GFS con los 5 conjuntos de medios, vaulting con custodia y evidencia, multistream y pools WORM (§8.4).

- La cinta entra como **un destino más** (§7.5): los trabajos, los manifiestos y la retención no conocen el tipo concreto de destino.

8.2 El driver genérico

El agente habla con la cinta a través de una interfaz única y secuencial (cargar/descargar por barcode o por ranura, rebobinar, buscar por marcas de archivo, escribir/leer en la posición actual, borrar, estado e inventario del cambiador):

- Con **hardware real** usa las herramientas estándar del sistema: la unidad (`/dev/nst*`) y el cambiador (`/dev/sg*`). Sin hardware, lo reporta con su motivo: **nunca inventa un equipo**.
- Con la **librería virtual en disco** (un directorio = una librería; cada cartucho = un subdirectorio con su barcode, sus datos y su índice de marcas) se desarrolla, se prueba y se demuestra **sin hardware**, con **la misma máquina de estados** que el hardware real: escribir o leer sin cargar = error; cargar un segundo cartucho sin descargar = error; cargar un barcode inexistente = error; buscar fuera de rango = error; escribir tras rebobinar trunca el resto.

Pool, catálogo y retención (simple):

- La política asigna un cartucho o un conjunto de barcodes (el pool). Si el cartucho cargado **no es del pool, la escritura falla en voz alta**: nada de escribir en el cartucho equivocado en silencio. La lectura sí cambia de cartucho: manda el catálogo.
- El **catálogo por barcode** (persistido) registra por cada objeto: clave, barcode, número de archivo en la cinta, tamaño, huella, fecha de escritura y fin de retención. **Es dato crítico**: sin catálogo, la cinta es una caja negra irrecuperable — respáldelo como tal.
- **Retención simple:** sobrescribir una clave con retención vigente = error; **reciclar** un cartucho (el único «borrado» que existe en un medio secuencial) con puntos vigentes = error. No existe borrado individual de un objeto en cinta, por la naturaleza del medio.
- **Restaurar desde cinta** = leer de vuelta al destino que se pida (staging a disco primero), verificable por huella contra el catálogo.

8.3 Cuando el genérico no basta: driver a la medida por canal seguro

Al detectar el equipo, el agente lo clasifica:

- El **genérico basta** si se cumplen las tres: (1) la unidad se expone como cinta SCSI estándar; (2) el cambiador, si existe, se expone como cambiador estándar operable con las herramientas del sistema; (3) el equipo no está en la tabla de los que exigen comandos propios del fabricante.
- Un equipo **no catalogado** no se declara «a la medida» automáticamente: queda como **«requiere evaluación»**.
- Si un equipo requiere driver propio, la consola presenta el aviso **«Se requiere un driver para tu equipo»** y el flujo es:

1. **Requerimiento (lo que viaja al proveedor):** el agente genera el archivo de requerimiento con **solo datos del equipo**: tipo, fabricante, modelo, ruta y motivo; el número de serie va **siempre redactado/parcial** (primeros y últimos caracteres, el resto oculto). Sin credenciales, sin datos del

cliente, sin contenido de respaldos. El archivo lleva su **huella**, que permite detectar cualquier alteración posterior. Si ningún equipo requiere driver a la medida, el comando **no genera un requerimiento vacío**: lo dice y termina con error.

2. **Construcción y entrega:** el proveedor construye el driver para ese equipo y lo entrega como **paquete firmado** por el **canal seguro**: un archivo con el manifiesto del driver (nombre, versión, equipos compatibles y la huella de **cada** archivo) y su **firma digital** separada.
3. **Instalación verificada — fail-closed, en este orden y todo obligatorio:**
    1. Debe existir la **llave pública del proveedor** configurada o embebida en el agente. Sin llave → rechazo.
    2. Estructura del paquete válida (solo archivos regulares, sin rutas que escapen, sin duplicados, con manifiesto y firma).
    3. **Firma válida** sobre los bytes exactos del manifiesto. Si no → rechazo.
    4. **Huellas exactas:** el conjunto de archivos es exactamente el declarado y cada huella/tamaño coincide. Si no → rechazo.
    5. **Compatibilidad:** alguna etiqueta de compatibilidad cubre el equipo detectado en ese equipo. Si no → rechazo.
    6. Registro: los archivos se copian al registro local de drivers, se actualiza el índice y el intento queda anotado en su registro de auditoría local.
7. Firma mala, hash alterado o equipo incompatible = **rechazo total**: no se instala nada, ni parcialmente, y el intento queda registrado con su motivo. Los drivers instalados se listan, se consultan sus intentos y se pueden **retirar** — todo registrado.

El estado de cinta del agente se expone siempre en su foto de estado (`tape`): disponibilidad y motivo, equipos detectados con su clasificación, barcode cargado, si se requiere driver a la medida (con su nota para la consola), los drivers instalados y los requerimientos pendientes. Sin hardware, la ficha muestra el motivo honesto («sin hardware de cinta en este host») en lugar de ceros.

> **Límite verificado — hardware real:** la validación contra una **librería LTO física** (coreografía real de carga/búsqueda, tiempos, limpieza de unidad, barcodes físicos) está **pendiente de disponer de la librería**. El código de hardware está probado con un ejecutor falso y con el analizador real de la salida del cambiador, y el extremo a extremo con objetos reales se verificó contra la librería virtual. No prometa soporte de ningún modelo concreto más allá de la clasificación de la tabla de equipos hasta que esa validación exista.

8.4 Infraestructura completa de cinta (v2)

Construida sobre §8.2–§8.3, sin reescribirlas:

- **Pools GFS con 5 conjuntos de medios:** cada día se clasifica en su conjunto — el nivel más alto manda: 1 de enero → **anual**; día 1 de trimestre → **trimestral**; día 1 de mes → **mensual**; domingo → **semanal**; resto → **diario**. Retenciones por defecto: diarios 7 días, semanales 5 semanas, mensuales 12 meses, trimestrales 2 años, anuales 7 años. El gestor registra los medios por barcode, asigna el medio del conjunto al escribir, extiende la retención del medio con cada punto, sella los medios llenos y localiza cada punto (qué barcode montar y en qué posición).

- **Vaulting con custodia y evidencia:** exportar un medio exige destino y actor, lo marca como **exportado** (irreciclable e inasignable mientras esté fuera) y registra el movimiento en un **registro de custodia append-only con cadena de hash**: quién entrega y quién recibe, cuándo, qué barcodes y a qué destino. Retorno y verificación en custodia son movimientos del mismo registro. Un registro manipulado se **rechaza al abrir**: la custodia es evidencia, no una nota.
- **Multistream:** varios flujos en paralelo, uno por unidad, cada uno con su propio manifiesto (unidad, barcode, posición, tamaño, huella) y **verificación al cierre** (relectura + huella antes de dar el flujo por bueno). Si un flujo falla, el trabajo devuelve error y su manifiesto lo reporta, pero los demás flujos quedan íntegros y verificados: no hay estado compartido entre unidades.
- **Pools WORM:** con WORM activo, la retención es **absoluta y fail-closed**: reciclar o reutilizar un medio antes de que venza su retención se rechaza **incluso forzándolo**. En pools no WORM, forzar es la vía deliberada del operador para reciclar anticipadamente (decisión humana registrada); sin forzar, la retención también se respeta.
- **Trazabilidad de la verificación:** un punto que vive en cinta es un manifiesto normal, así que la misma verificación en sandbox de §7.6 lo cubre sin código nuevo: el operador resuelve el artefacto desde la cinta (localizar el punto → montar el barcode → leer) y verifica igual que cualquier otro destino.

---

9. Recuperación ante desastres (DRP)

9.1 El runbook: reconstruir es re-aplicar

Un runbook es una receta especial versionada (`kind: Runbook`) que describe el orden canónico de reconstrucción:

1. **Red:** redes y espacios de nombres, con sus políticas.
2. **Volúmenes:** crear los volúmenes vacíos con los mismos nombres y clases.
3. **Datos:** restaurar los snapshots de volúmenes y, para las bases, restaurar desde el módulo de respaldos (§7) eligiendo el punto — incluido un punto PITR (§7.4).
4. **Aplicaciones:** desplegar los servicios en orden topológico según sus dependencias, esperando a que cada capa responda sana antes de seguir.

Como la receta siempre refleja la realidad (reconciliación + captura inversa, §5), recuperarse no es improvisar: es re-aplicar. Los objetivos de diseño por nivel:

Nivel	RPO objetivo	RTO objetivo	Notas
Comercial Esencial	24 h	4 h	Snapshots diarios, restore manual asistido
Comercial Profesional	1 h	1 h	Snapshots horarios, runbook semi-automático
Gobierno	Contractual (diseño: 15 min)	Contractual (diseño: 30 min)	Semi-automático o autónomo por política (§9.3) + drills mensuales evidenciados

Sin promesa de pérdida cero: el diseño ofrece RPO por snapshots frecuentes y por CDC (§7.9), no replicación síncrona.

9.2 Ejecución semi-automática (línea base v1/v1.2)

La detección y la preparación son automáticas (punto verificado más reciente, chequeos de destino, plan ordenado). La **ejecución exige autorización humana**, por dos canales:

- **Consola:** botón **Ejecutar DRP** con doble confirmación. Queda registrado quién autorizó.
- **Llamada de voz:** si nadie responde en la consola, el sistema **llama al responsable de guardia** (números pre-registrados): el agente de voz explica qué se detectó y qué se recuperará, y pide la autorización por voz («diga autorizo») o por teclado (tecla 1).

Reglas de seguridad no negociables de la voz:

- La llamada sola **no basta**: número pre-registrado **+ PIN** hablado o por teclado.
- La grabación y la transcripción quedan en el registro de auditoría como evidencia.
- El webhook que entrega el resultado de la llamada verifica su **firma en modo fail-closed**: sin firma válida, no se acepta ninguna autorización. (Esta fue una brecha conocida de la plataforma heredada — webhook sin firma que aceptaba en abierto — y está registrada como requisito a verificar antes de usar la voz para DRP real.)
- Si nadie de la cadena contesta tras los intentos configurados: **no se ejecuta nada con datos**; se escala y se alerta por todos los canales (consola, Teams, correo, SMS y las integraciones de §13).

Separación de funciones (SoD): quien configuró un runbook **no puede ser su único aprobador**. La ejecución exige un aprobador distinto del configurador.

9.3 Políticas autónomas v2: autonomía gobernada

En v2, cada workload o nivel puede declarar su **nivel de autonomía**. El humano deja de ser el cuello de botella de las 3 a.m. y pasa a ser el **legislador**: define, firma y puede revocar las reglas dentro de las cuales el sistema decide solo.

Nivel	Comportamiento
`manual`	Como la línea base: nada se ejecuta sin aprobación humana
`aprobación diferida`	Se ejecuta si **nadie lo veta** dentro de la ventana configurada (N minutos). El veto se ejerce desde la consola o desde el móvil (§17)
`autónomo acotado`	Se ejecuta sin humano, **solo** dentro de los guardarraíles de la política

El techo lo fija la instalación, no la receta. El nivel efectivo es el menor entre el de la política y el máximo permitido por la instalación. Sin configuración de guardarraíles de instalación, el arranque es `manual` (fail-closed).

9.3.1 Crear y activar una política autónoma

1. En **DRP → Políticas**, cree la política: workloads cubiertos (**alcance cerrado**: fuera de esa lista, la política no existe), nivel, ventana horaria local, RPO máximo aceptable antes de disparar, ventana de veto (si aplica) y la exigencia de snapshot previo.
2. Guardar crea una **versión nueva en borrador, sin firmas**. Editar una política activa hace lo mismo: la versión vigente no se toca hasta que la nueva se active.
3. **Activación con doble firma (SoD reforzada):** activar exige la firma de un administrador **distinto** de quien la creó. Sin segunda firma, la política queda en borrador inactivo y el evaluador la trata como **inexistente**.
4. La política activada, su techo y el interruptor de apagado quedan en auditoría con cadena de hash.

9.3.2 Guardarraíles: qué comprueba el sistema antes de actuar solo

Antes de cualquier acción autónoma, el evaluador verifica **todos** los guardarraíles. Basta que uno falle para que el caso caiga a manual (o a rollback, en la verificación posterior):

Guardarraíl	Si falla
Interruptor de apagado (kill switch) de la instalación activado	Manual: cero autonomía en toda la instalación
Política no activa o sin doble firma válida	Manual
Nivel pedido por encima del techo de la instalación	Se **acota** al techo (no se rechaza: se degrada al nivel permitido)
Workload fuera del alcance cerrado de la política	Manual
Fuera de la ventana horaria local	Manual
RPO observado mayor que el máximo de la política, o **desconocido**	Manual
Snapshot previo ausente (cuando la política lo exige)	Manual
Verificación posterior a la restauración fallida o **ausente**	**Rollback** automático

Cada decisión se persiste con el resultado **por guardarraíl** y encadenada por hash: se puede reconstruir después por qué el sistema actuó, esperó o no actuó.

Doble verificación de la decisión. La consola decide (allí viven la política firmada, el techo, el kill switch, la ventana de veto y la auditoría) y el **agente re-verifica** la orden autónoma con la misma función pura antes de ejecutar: si su re-verificación no reproduce «ejecutar», el agente **se niega** y el caso cae a manual. Hacen falta consola y agente de acuerdo para que algo se ejecute sin humanos.

Ventana de veto. En `aprobación diferida`, la evaluación queda en espera hasta el fin de la ventana, como estado persistido: vetar dentro de la ventana la detiene; vencida sin veto, se libera a ejecutar.

Verificación posterior y rollback. Tras restaurar, el sistema exige **doble verificación**: salud **y** consistencia. Si cualquiera falla — o no consta — la evaluación queda marcada como «rollback requerido» y se dispara el rollback. Una verificación ausente cuenta como fallida: fail-closed.

Kill switch. En **DRP → Guardarraíles de la instalación**, un administrador puede activar el interruptor global: efecto inmediato, cero acciones autónomas en toda la instalación hasta desactivarlo. Los flujos manuales no se ven afectados. Activarlo y desactivarlo queda en auditoría.

9.4 Drills (simulacros) y su informe

- **Drill programado** (mensual por defecto, configurable): el sistema levanta un entorno sandbox, restaura la última copia de cada workload marcado para DRP, corre sus comprobaciones de salud y emite el **reporte del drill**: éxito/fracaso por workload y tiempos medidos contra el RTO/RPO del nivel. Un restore que nunca se prueba es un restore que no existe: el reporte es su evidencia auditable.
- **Drill bajo la política autónoma (v2):** el drill corre por el mismo pipeline (sandbox, sin tocar producción) y su reporte añade **qué habría hecho la política** ante la misma señal frente a lo ejecutado, con marca de divergencia si difieren. Así la autonomía se prueba continuamente y no debuta en el desastre real.
- Los informes de drill pertenecen a la familia de informes firmables y con «Ver pruebas» (§12).

---

10. Licenciamiento y función de pago de VMs

10.1 El modelo en una página

- **Unidad = agente.** Cada agente enrolado consume un cupo; los contenedores por agente son ilimitados.
- **La asignación manda, y vive fuera de Abzu.** Qué cuenta/tenant tiene qué plataforma, plan, cupo y funciones lo gobierna el **panel de licencias y plataformas del proveedor** de la plataforma. Abzu **no crea un registro de asignación paralelo**: valida contra lo asignado allí.
- **El punto técnico de validación** es el servicio de licencias : registro automático de instalaciones, latido de cupo, emisión alimentada por la asignación del proveedor y licencias offline firmadas para gobierno. Su portal web conserva compras y reportes y muestra el estado por identificador de instalación (vista de solo lectura; las operaciones sensibles exigen el token de la instalación, nunca el solo identificador).
- **Bloqueo suave, siempre.** Vencida una prueba o una suscripción fuera de su gracia, o agotado el cupo: no entran servidores nuevos ni se despliega nada nuevo, pero **nada de lo que ya corre se apaga ni se toca**.

10.2 Registro zero-touch y prueba

1. Al instalarse, la instalación se **registra sola** (sin intervención del proveedor): aporta nombre, edición (`comercial` / `gobierno`), contacto y su huella, y recibe su identificador de instalación y un

token que se entrega una sola vez (el servicio guarda solo su huella; preséntelo en cada latido como credencial).

2. El registro abre una **prueba de 30 días / 3 servidores**, sin tarjeta.
3. En cada latido, la instalación reporta sus servidores activos. Respuesta: estado, cupo, usados/disponibles, días restantes y si puede añadir servidores y desplegar.
4. **Servidor que deja de reportarse:** su cupo se libera tras **7 días de gracia**. Si vuelve a reportarse dentro de la gracia, se reactiva y la gracia se cancela.
5. **Cupo agotado:** lo existente sigue; los servidores nuevos se rechazan **suavemente** (quedan listados como rechazados en la respuesta, con el resto operando) hasta ampliar el cupo.
6. **Suscripción vencida:** 7 días de gracia de licencia y después estado vencido — siempre bloqueo suave. La licencia **perpetua nunca vence** (días restantes = sin vencimiento); que venza el soporte opcional **no** la bloquea.

Estados de la licencia que verá en la consola y en el portal: `active` (vigente) · `trial` (en prueba) · `grace` (vencida hace ≤ 7 días) · `soft_blocked` (prueba vencida) · `expired` (suscripción vencida fuera de gracia).

10.3 Solicitar ampliación y cómo se activa

1. Desde la consola o el portal, registre una **solicitud de compra/ampliación** (tipo, cupo pedido, contacto, notas). La solicitud **solo se registra: no cobra** ni pide datos de pago.
2. La activación llega cuando el pago (comercial) o la adjudicación (gobierno) se confirma en el panel del proveedor: su confirmación alimenta la emisión técnica — por la vía interna gobernada o por el webhook de pago con **firma HMAC fail-closed** e **idempotente** (un evento duplicado no re-emite) — y la licencia queda emitida sin intervención manual.
3. Sin secreto de webhook configurado, ese endpoint queda **deshabilitado** (no «abierto»): fail-closed también en la emisión.

10.4 Función de pago de VMs/hipervisores

La gestión de VMs/hipervisores (§4.4) se activa solo si la licencia **asignada por el proveedor** la contempla (función `vm_hypervisor`). La consola valida esa asignación contra el punto técnico y el agente vuelve a aplicar el gate sobre las acciones. Estados que verá:

Estado	Qué ve y qué puede hacer
Disponible (`entitled`)	La asignación contempla la función y la licencia está vigente (o en gracia de licencia: lo existente sigue, marcado como degradado). Descubrimiento y acciones permitidos. En la instalación de referencia del piloto gobierno, la función sale **disponible** porque su asignación la contempla — no por ningún trato especial por nombre: manda la asignación
Prueba (`trial`)	**Todo funciona, también las VMs.** La consola muestra el **modal de conversión**: informa de que la función es de pago y ofrece activarla, **sin bloquear su uso**. La prueba no tiene paredes; la conversión se gana mostrando el producto completo

Estado	Qué ve y qué puede hacer
Sin función (`none`) — bloqueo suave	La sección **se ve** (con su aviso de función de pago), pero las **acciones se deniegan** (en web: redirección con aviso; en API: `403` con código de entitlement requerido). El resto del producto sigue operando; jamás se tocan datos del cliente
Validación imposible	Fail-closed **con la gracia definida**: dentro de la gracia se conserva el último estado conocido, marcado como degradado; fuera de la gracia o sin foto válida guardada, la función queda en «sin función». El agente aplica el mismo criterio con su propia caché y su gracia de entitlement

Gobierno paga precio completo. En edición gobierno no existe precio de prueba ni descuento para esta función: su conversión es siempre a **precio completo** (perpetua por agente + soporte anual), desde la adjudicación. En comercial, la prueba convierte por activación y la ausencia de función, por compra.

10.5 Licencia offline (gobierno sin red)

Para instalaciones sin salida a internet, la licencia viaja como un **archivo firmado** (firma digital ed25519):

1. El emisor genera el archivo para **esa** instalación: identificador de la licencia, identificador de instalación, titular, edición, tipo (`perpetua` / `suscripcion`), cupo de servidores, fechas de emisión, vencimiento (vacío en perpetua: **nunca vence**) y de fin de soporte.
2. La instalación lo importa desde la consola. La consola **verifica localmente, sin red**: firma válida contra la llave pública del emisor que trae incorporada, identificador de instalación coincidente con el suyo y vigencia. Cualquier alteración del archivo (cupo, identificador, tipo) **invalida la firma** y se rechaza: fail-closed ante firma inválida, algoritmo distinto, versión no soportada, cupo inválido o suscripción sin vencimiento o ya vencida.
3. La consola revalida el archivo **al arrancar y cada 24 horas**, siempre sin llamadas externas. La llave privada del emisor **nunca** se distribuye.

10.6 Licenciamiento federado

En una federación (§15), el conteo se agrega hacia el servicio de licencias: cada instalación produce **reportes firmados** de su uso y la central los agrega — en gobierno, como **agregado offline firmado**, verificable con llaves públicas aunque la central esté en failover. El portal de licencias gana además el autoservicio federado sobre la **misma** asignación del proveedor: compra de cupo por sitio, reportes agregados por institución y alertas de capacidad antes del agotamiento. Nunca sobre un registro paralelo.

---

11. Seguridad, roles y cumplimiento

11.1 RBAC fino con denegación por defecto

Roles: **operador**, **aprobador**, **auditor** y **solo lectura**. Cada endpoint de la consola y de la API interna se mapea a una **acción** (`view`, `operate`, `approve`, `audit`, `destroy`, `debug`) y cada acción a los roles que pueden ejecutarla. La denegación es por defecto en **tres niveles**:

1. Un endpoint **sin acción explícita** en el mapa → **403**.
2. Un usuario **sin rol resoluble** → **403**.
3. Una acción desconocida → **403**.

Toda denegación queda en auditoría. La resolución del rol sigue este orden: rol explícito del usuario → rol de su sesión de SSO/LDAP → asignación por correo configurada en la instalación → mapa heredado de su equipo. Un rol explícito inválido **deniega**; no degrada en silencio a un rol menor.

Algunas referencias de uso: ver = los cuatro roles según su alcance; operar = operador y aprobador; aprobar = aprobador; auditar y exportar evidencia = auditor (y administrador); destruir y depurar = solo con los controles adicionales de §11.2 y §11.4.

11.2 Autenticación adaptativa (step-up)

Ante un **contexto anómalo**, la consola exige una **re-autenticación reforzada** antes de las acciones sensibles. Señales (cualquiera dispara; cada una aporta a la puntuación de riesgo que queda registrada): **IP nueva**, **dispositivo nuevo** (huella del navegador) y **horario inusual** (fuera de la ventana configurada de la instalación; por defecto 06:00–22:00 en la zona horaria de referencia de la instalación).

- La primera petición con sesión establece su línea base de contexto conocido; completar un step-up incorpora el contexto vigente a esa línea base.
- **Acciones sensibles con step-up:** destruir recetas, crear o decidir recertificaciones y activar la consola de depuración.
- El step-up pide su **contraseña** y, si su cuenta tiene segundo factor confirmado, también el **código TOTP**. La marca de step-up vale **10 minutos** (configurable).
- Sin marca válida, la acción sensible se deniega — en web, redirigiéndole al formulario de step-up conservando su destino; en API, con `403` y código `step_up_required` — **sin expulsarle**: lo no sensible sigue funcionando. Si el proveedor del segundo factor no está disponible y su cuenta lo exige, también se deniega: fail-closed.

11.3 La cuenta de soporte del proveedor (acceso total registrado)

Por regla permanente del proveedor, la cuenta del centro de operaciones de seguridad del proveedor es **super-administradora del producto**: acceso total a la consola, a todos los tenants e instalaciones y a todas las funciones — incluidas las de pago y las de gobierno — sin gates de licencia ni de rol que la limiten, y destinataria de las alertas críticas (DRP, licenciamiento, seguridad).

Lo que esta regla **sí** exige, igual que a cualquier cuenta: autenticarse como esa cuenta y superar el step-up adaptativo cuando aplique (eso verifica **identidad**, no limita rol ni licencia). Y cada acceso suyo por el control de roles queda en el registro de auditoría **citando la regla**, como cualquier otro evento. Usted puede auditar esos accesos en su propia bitácora: el acceso total es también acceso totalmente registrado.

11.4 Consola de depuración gobernada

La consola de depuración es la única superficie que se acerca a una ejecución directa, y por eso está gobernada:

- **Desactivada por defecto** en la configuración de la instalación. Apagada en configuración, cualquier sesión viva **muere**.
- **Aprobación dual:** la solicitan y la aprueban **dos aprobadores distintos**, y nunca el propio solicitante (separación de funciones).
- **Step-up obligatorio** para activarla (§11.2).
- **Sesión efímera:** 15 minutos por defecto, sin renovación silenciosa. Su estado vive en memoria volátil: un reinicio la apaga, no la revive.
- **Grabación íntegra obligatoria:** sin grabación no hay sesión. Solicitud, aprobaciones, inicio y fin quedan en la transcripción y en el registro de auditoría.

Si su instalación no la ha habilitado deliberadamente con este procedimiento, no existe: no hay una «consola oculta» alternativa.

11.5 Recertificación de accesos

Campañas periódicas (semestrales) de revisión de quién tiene acceso a qué, con evidencia:

1. Un administrador crea la **campaña** de recertificación.
2. Cada acceso se decide individualmente: **certificado** (sigue) o **revocado** (se retira). Nadie puede decidir sobre **su propio** acceso (SoD), y crear/decidir campañas son acciones sensibles con step-up (§11.2).
3. La campaña y sus decisiones se **exportan como evidencia** para auditoría.

11.6 Escaneo de imágenes, Fail2Ban y otras protecciones

- **Escaneo de imágenes en la admisión (anti-CVE):** antes de aplicar una receta, sus imágenes se escanean en busca de vulnerabilidades conocidas. Una imagen en **cuarentena** según la política (por defecto, hallazgos críticos corregibles) **bloquea** la aplicación sin tocar el motor. El escáner es una pieza intercambiable tras una interfaz.
- **Fail2Ban:** el agente lee el estado **real** de Fail2Ban del host (jaulas — jails —, fallos y baneos actuales con sus IP) y lo expone en la foto de estado del agente, **siempre presente**: si el host no tiene Fail2Ban, el campo dice «no disponible» con su motivo, para distinguir «sin Fail2Ban» de «sin datos». La receta puede **declarar** las jaulas esperadas en el host (p. ej. protección de SSH y de puertos sensibles); el agente las contrasta con la realidad. La **operación diaria** de Fail2Ban

(baneos, listas de permitidos, decisiones) vive en su consola propia enlazada desde la ficha del agente — el mismo Fail2Ban que el agente reporta — con su propia autenticación; Abzu la **enlaza, no la reescribe ni la embebe con credenciales**.

- **Actualizaciones y artefactos:** releases del agente firmados con verificación obligatoria y rollback automático (§2.7); lista de materiales de software (SBOM) publicada por release; los bundles offline se verifican por firma antes de aplicarse (§6.5, §10.5).

> **Nota honesta — segundo factor en cuentas locales:** vía SSO/OIDC, el segundo factor lo aporta el proveedor de identidad de su institución. Para las **cuentas locales** de la instalación gobierno, la obligatoriedad del segundo factor para administradores es un requisito registrado como pendiente en la línea de endurecimiento del producto: este manual no lo reclama como disponible para cuentas locales hasta que su instalación lo evidencie. Planifique las cuentas locales de administrador en consecuencia (contraseñas fuertes, step-up activo y recertificación al día).

11.7 Auditoría, cadena de hash y exportación de evidencia por marco

Registro de auditoría. Todo evento de seguridad y de operación — enrolamientos, inicios de sesión (correctos y fallidos), accesos sensibles, despliegues, convergencias, capturas, aprobaciones y decisiones de DRP (con su canal: consola, voz o móvil), cambios de licencia, recertificaciones, decisiones denegadas por RBAC y accesos de la cuenta de soporte — se registra con fecha y hora, actor, tenant y resultado. En la forma gobierno el registro es 100 % local y **exportable al SIEM** de la institución (syslog/CEF). Los cambios sensibles se guardan cifrados en reposo.

Cadena de hash. Varias cadenas del sistema enlazan cada entrada con la huella de la anterior (auditoría de decisiones, catálogo de copias offline, custodia de cintas, evaluaciones de DRP): alterar, borrar en medio o reordenar entradas **rompe la cadena y se detecta al verificarla**; un registro manipulado se rechaza al abrirlo.

Exportación de evidencia por marco. Desde **Evidencia / Cumplimiento**, un administrador o auditor genera el paquete de evidencia de su instalación: los hechos ya registrados (auditoría, inventario, recertificación, revisiones de recetas) **etiquetados por control del marco que elija**, en manifiesto JSON con la huella SHA-256 de cada hecho y un índice CSV hecho × control, re-verificable por un auditor externo sin tener que confiar en la consola. El paquete se completa con el whitepaper de arquitectura de seguridad y el SBOM.

La regla de mercados (no negociable):

Mercado	Marco
República Dominicana (gobierno)	**NORTIC**
Estados Unidos	**NIST** (SP 800-53 / CSF)
Resto del mundo	**ISO/IEC 27001**

NORTIC es **solo** República Dominicana y **nunca se comercializa fuera de RD**. Un mismo hecho, tres lecturas: el etiquetado cambia, la evidencia es la misma.

Alcance honesto del cumplimiento. El sistema de gestión — políticas, riesgos, recursos humanos, seguridad física, el centro de operaciones — es de **su institución** en cualquier mercado. Abzu lo **evidencia**, no lo sustituye: un control sin hechos registrados sale en el paquete con la lista vacía, no con evidencia fabricada. Dígalo así desde el primer día ante cualquier auditor: es lo que hace creíble todo lo demás.

---

12. Firma de informes y validación «Ver pruebas»

12.1 El módulo interno de firma

Los entregables del producto — informes de auditoría, de validación operativa, de respaldos y de DRP — se entregan **firmados y verificables**. La firma se calcula con un **módulo interno residente en la plataforma**: la firma vive dentro de su instalación y **no se consume llamando a ningún servicio externo**. Funciona igual en una instalación gobierno sin salida a internet o aislada (air-gapped).

La firma es digital (algoritmo ed25519) sobre el documento y su contenido exacto: documento + huella de su contenido + firmante + momento + firma. Alterar el documento cambia su huella y la verificación falla; alterar el registro de la firma también.

12.2 Cargar los datos de firma en Configuraciones

Cada instalación carga **sus propios** datos de firma. Solo administradores de la instalación:

1. Vaya a **Configuraciones → Firma de informes**.
2. Complete la **identidad del firmante**: nombre, cargo/rol, correo y organización. Es la identidad que constará en cada firma de esta instalación.
3. Aporte la llave de firma, por una de estas dos vías:
    1. **Generar:** la plataforma genera un par de llaves nuevo **dentro de su instalación**. La llave privada queda guardada **cifrada en reposo**; la pública y su huella quedan visibles (son públicas). Generar sustituye la identidad anterior: **una identidad activa por instalación**.
    2. **Importar:** pegue la llave privada propia de su instalación (en el formato que pide el formulario). Se valida **antes** de guardarla derivando su llave pública y su huella; si no es válida, se rechaza y no se guarda nada.
3. Guarde. La llave privada **jamás se vuelve a mostrar**: ni en esta pantalla ni en ninguna otra. Actualizar los datos visibles del firmante (nombre, cargo, correo, organización) no toca las llaves.
4. Sin identidad de firma activa, **no se firma**: el intento de firmar un informe sin identidad configurada se rechaza con su mensaje (fail-closed). Configure la firma **antes** de su primera entrega.

12.3 Firmar y entregar un informe. «Ver pruebas»

Cada corrida de validación (operativa, de respaldos o de DRP) tiene su informe web:

1. En **Validación**, cada corrida de la lista muestra su tipo, alcance, fecha y resultado global, y la acción **«Ver pruebas»**.

2. **«Ver pruebas»** abre el informe de esa corrida: encabezado (instalación/tenant, fecha, alcance, resultado global), la **lista de pruebas aplicadas con su resultado** (superada / fallida / omitida / error, con su detalle) y una **tira de imágenes** con la **captura real** de cada prueba que la genera (la misma captura que produce la verificación de respaldos, §7.6). Sin imágenes de relleno: la prueba que no genera captura muestra su **evidencia textual** en el mismo informe.
3. Las corridas llegan a esta lista por dos vías: registro manual de una corrida con sus pruebas (sin capturas falsas: lo que se registra a mano trae su evidencia textual) o **importación del informe de salud** que genera la verificación del agente, que se convierte automáticamente en corrida con sus pruebas.
4. **Firmar:** con la identidad de §12.2 cargada, un administrador elige **Firmar** en el informe. La plataforma firma el contenido canónico de la corrida y **registra la firma** (documento, huella, firmante, momento).
5. **Regla de entrega:** un informe **sin firma registrada y verificada no cuenta como entregado**. La lista de corridas le muestra, por cada una, si está entregada — y esa comprobación verifica la firma contra el contenido actual: si el contenido cambió después de firmar, ya no consta como entregada.

12.4 Verificar un informe recibido

Quien recibe un informe puede comprobarlo sin acceso a la consola:

1. Abra la **página pública de verificación** del informe (la ruta de verificación que acompaña a la firma, con limitación de velocidad).
2. La página muestra los datos registrados de la firma y permite **comprobar** el documento: se recalcula su huella y se verifica la firma contra la llave pública de la instalación firmante.
3. Resultado posible: **válida** (el documento es exactamente el firmado, por ese firmante, en ese momento) o **no válida** (el documento, su huella o la firma no coinciden: no lo acepte como entregado y escale por su canal habitual).

---

13. Integraciones y notificaciones

13.1 Canales

Configure en **Integraciones** los canales de **su** institución — los nombres que siguen son integraciones que usted conecta con sus propias cuentas, no la voz del producto:

Canal	Qué recibe	Configuración por tenant
Correo	Alertas e informes por el correo de su institución (en gobierno, vía su Exchange Online)	Remitente y destinatarios de su instalación
Microsoft Teams	Alertas a un canal de su Teams (webhook entrante suyo)	URL del webhook por severidad si lo desea
Slack		

Canal	Qué recibe	Configuración por tenant
	Alertas a canales suyos, **por severidad**	Una URL de webhook por defecto y, opcionalmente, una por severidad
PagerDuty	La alerta **abre y cierra** el incidente con la guardia correcta	Clave de enrutado por defecto y, opcionalmente, por severidad. El acuse (ack) desde PagerDuty se refleja de vuelta en la consola
ServiceNow	Tickets de cambio/incidente **enlazados** al evento y a la revisión de receta que lo causó	URL de su instancia y credenciales suyas. El cierre del incidente enlaza el informe post-incidente
Webhooks genéricos firmados	Cualquier sistema suyo suscrito a los eventos del producto	Ver §13.2

Cada canal se activa por separado, con su **severidad mínima** (información, éxito, aviso, error, crítica): por debajo de ella, el canal no recibe nada. Los secretos de cada canal (URL con token, claves, contraseñas) se guardan como secretos: el formulario **solo los sobrescribe cuando usted escribe un valor nuevo** y **nunca los muestra**.

13.2 Webhooks firmados y suscripciones

1. En **Integraciones → Suscripciones**, cree una suscripción: nombre, URL de destino y los **tipos de evento** que quiere recibir (despliegue, drift, respaldo, DRP, licencia, entre los que lista la pantalla) con su severidad.
2. Cada entrega va **firmada (HMAC)** con el secreto de la suscripción: su sistema debe verificar la firma sobre el cuerpo crudo antes de aceptar el evento — es el mismo estándar fail-closed que se exige a cualquier webhook que autoriza acciones (§9.2). Una entrega sin firma válida debe rechazarse, también de su lado.
3. Las entregas fallidas se **reintentan** con la política de reintentos del producto y quedan registradas: en **Integraciones** ve las últimas entregas con su resultado, y los **enlaces externos** que las integraciones bidireccionales han creado (incidente/ticket ↔ evento ↔ revisión de receta).

13.3 Qué eventos salen

Despliegues correctos y fallidos, drift detectado, respaldos fallidos y verificaciones con resultado, eventos de DRP y de drills, licencias próximas a vencer o en gracia, y alertas de observabilidad (§16). El catálogo exacto de tipos lo lista la propia pantalla de suscripciones de su instalación: suscríbase solo a lo que su sistema vaya a atender — una alerta que nadie atiende es peor que ninguna, porque entrena a ignorarlas todas.

---

14. API pública, SDKs y Terraform

14.1 API pública versionada

Además de la API interna con la que opera la propia consola, el producto expone una **API pública versionada** para clientes e integradores, en su propio prefijo (`/api/public/v1`), separada de la interna:

- **Autenticación:** token de API del tenant (portador, bearer). El tenant se deriva **siempre** del token: cualquier identificador de tenant que el cliente envíe se **ignora**.
- **Alcance:** agentes, recetas, despliegues, respaldos y políticas de respaldo, políticas de DRP, enrolamientos de agentes y estado de licencia — con las mismas garantías que la consola. La especificación normativa es el documento OpenAPI del producto.
- **Contrato de respuestas:** éxito como `{"data": ...}`; error como `{"error": {"code": "<clave estable>", "message": "..."}}`. Programe contra `code`, nunca contra el texto del mensaje (que es apoyo, en su idioma). Códigos: `unauthenticated` (401), `forbidden` (403), `not_found` (404 — **también** para recursos de otro tenant: no se revela su existencia), `validation_failed` (422, con el detalle por campo), `rate_limited` (429).
- **Garantías: cada llamada queda en auditoría**; hay **límite de velocidad por tenant**; y la superficie es de **denegación por defecto** — solo existen las rutas publicadas en el OpenAPI.
- **Capacidades por token:** la lectura es la base; la escritura (p. ej. crear/actualizar recetas, que nacen con revisión 1 y la incrementan al actualizar) exige la capacidad de escritura en el token. Los despliegues y los respaldos programados se exponen en lectura por esta API.

14.2 SDKs oficiales

Hay SDKs oficiales en **Go** y **TypeScript** (Python según demanda), **generados desde el mismo OpenAPI** para que no diverjan de la API. Úselos en vez de construir clientes a mano: heredan el contrato de sobres, los códigos de error y la derivación del tenant por token.

14.3 Proveedor Terraform

Para equipos que ya viven en infraestructura como código, el proveedor Terraform trabaja sobre la API pública v1 y comparte el mismo cliente tipado del SDK Go:

Recurso	Operaciones	Nota
`abzu_recipe`	Crear / leer / actualizar	La API pública no expone borrado de recetas: destruir el recurso en Terraform solo lo retira del estado
`abzu_agent_enrollment`	Crear (emite el token) / destruir (lo **revoca**)	El `token` es sensible y solo existe tras la creación
`abzu_backup_policy`	CRUD completo	—
`abzu_drp_policy`	CRUD completo	

Recurso	Operaciones	Nota
		Sujeta a las mismas reglas de doble firma para activarse (§9.3)

> **Límite verificado — distribución:** el proveedor **no se publica en ningún registro** de proveedores. Se usa **en local** con el mecanismo de sustitución de desarrollo de Terraform (`dev_overrides`): compile el binario, apunte su configuración de CLI de Terraform a esa carpeta y configure el proveedor con variables de entorno (endpoint y token) — nunca credenciales en el código. Con `dev_overrides` no hace falta inicializar desde un registro.

14.4 GitOps: interoperar, no incrustar (decisión documentada)

La evaluación comprometida en la hoja de ruta se ejecutó y su decisión es: **interoperar, no incrustar** un motor GitOps de terceros dentro del producto. Razones, en corto: la forma gobierno debe funcionar sin Kubernetes, sin red y en un solo servidor; el agente **ya es** el reconciliador (estado deseado versionado + reconciliación continua + captura inversa es exactamente el patrón GitOps); y un segundo bucle de reconciliación sobre los mismos workloads crearía conflictos de propiedad imposibles de auditar.

En la práctica: quien ya opere con un sistema GitOps consume las recetas compiladas **desde fuera** — por la API pública (las recetas exponen su número de revisión para detectar cambios), por los SDKs o por el proveedor Terraform — y esos workloads quedan fuera del ámbito del agente **por decisión explícita del operador**, nunca en disputa automática entre dos bucles. Una futura sincronización desde un repositorio Git externo, si llega, será **unidireccional** (Git → consola, con la consola en solo lectura para esas recetas): la doble fuente de verdad escribiendo en ambos lados es el fallo clásico de estas integraciones y no se va a construir.

---

15. Federación de consolas multi-sitio

15.1 Qué federa y qué no

Una consola **central** (p. ej. la de una institución coordinadora) ve y gobierna — según permisos — las consolas de sitio/institución federadas, **sin que los agentes cambien su phone-home local**: cada agente sigue hablando solo con su consola de sitio.

La federación intercambia **metadatos firmados**: inventario contado/agregado, estado de licencias, salud y políticas. **Nunca** secretos ni datos de workloads (principio de datos en casa del cliente). La sincronización es **oportunista, con reintentos**: si un sitio no sincroniza, la central lo muestra con su marca de última sincronización y su salud como **obsoleta** visible — no como si estuviera al día.

Cada sitio sigue operando **autónomo** aunque la central o la federación caigan: pierde la vista agregada, no la operación.

15.2 Operaciones de federación

1. **Registrar un sitio:** en **Federación**, dé de alta la consola de sitio con su identidad. La relación se establece sobre metadatos firmados entre consolas; no se comparten credenciales de agentes ni de workloads.
2. **Sincronizar:** la sincronización corre oportunistamente; también puede dispararla por sitio desde la misma pantalla. Revise la marca de última sync y la salud por sitio antes de tomar decisiones sobre datos agregados.
3. **Destino de respaldo en otro sitio:** defina un **destino federado** (el almacenamiento de objetos del sitio alterno, con credenciales **referenciadas**, reutilizando el destino de objetos existente) y úselo como segundo destino de sus políticas (§7.5): el respaldo de un datacenter vive en el otro.
4. **Plan DR entre datacenters:** declare el **plan** (sitio alterno + destino coherente con ese sitio; el modelo lo valida) para que un runbook pueda reconstruir en el sitio alterno desde ese respaldo federado.

15.3 Alta disponibilidad de la propia consola: decisión

Decisión de v2: activo-pasivo con failover probado. La consola central es una vista y un punto de administración agregado: un sitio primario la sirve y un sitio alterno mantiene su réplica en espera (base restaurable desde los respaldos federados) y asume el servicio con un procedimiento de failover **ensayado periódicamente y registrado como cualquier drill** — reconstruir la central en el alterno desde el respaldo federado y verificar la vista agregada.

El activo-activo multi-sitio queda como **evaluación futura, no como función de v2**: exigiría replicación multi-maestro y resolución de conflictos de escritura sobre objetos operativos que hoy no tienen semántica de fusión, con riesgo de doble verdad entre datacenters, a cambio de un RTO ≈ 0 que ningún escenario actual exige (los sitios, que son quienes operan, ya son autónomos). **No prometa activo-activo.**

Consecuencias prácticas: la sincronización no es en tiempo real (es oportunista y así se muestra), el licenciamiento federado (§10.6) vale también durante un failover porque se verifica con llaves públicas, y ningún procedimiento de su operación debe asumir escrituras cruzadas entre consolas.

---

16. Observabilidad: anomalías, predicciones y KPIs

La observabilidad avanzada trabaja sobre las series que el propio sistema ya produce (latido cada 30 s y fotos de estado con la misma cadencia). Tres reglas la gobiernan: **sin caja negra** (cada alerta explica qué se desvió y cuánto), **sin datos no hay cifra** (ausencia explícita, nunca un cero maquillado) y **nada se aplica solo** (toda recomendación es un borrador con aprobación).

16.1 Detección de anomalías sobre la línea base

- Por cada serie (sujeto × métrica: CPU, memoria, disco, red, reinicios, latencia de comprobaciones de salud, profundidad de cola — por workload o por nodo), la **línea base** es la media y la desviación de la ventana histórica previa (por defecto: 7 días).

- Una muestra es anómala si se desvía al menos **3 desviaciones** de la media. Cada evento de anomalía lleva: métrica, valor observado, media, desviación, dirección, ventana y número de muestras, más una **explicación en lenguaje claro generada desde esos mismos datos**. El evento alimenta el mismo registro de auditoría y los canales de alerta (§13, §17).
- **Fiabilidad declarada:** si la línea base no es fiable (pocas muestras — por defecto menos de 20 — o poca cobertura — menos de 24 horas —, o serie constante sin desviación), **no se emite ninguna anomalía** y el propio resultado explica por qué. Una línea base fiable necesita, con los valores por defecto, **del orden de 7 días de historial** por serie. Y una serie con estacionalidad fuerte (día/noche) mezcla regímenes en una línea base global: téngalo en cuenta al interpretar.

16.2 Predicciones concretas (alcance honesto)

No se promete «inteligencia que lo predice todo»: se prometen predicciones concretas, medibles y con su base publicada. Cada predicción sellada en auditoría queda fechada con su base, para poder medir su acierto después comparando lo proyectado con lo que ocurrió.

Predicción	Cómo se calcula	Cuándo calla
Disco que se llena	Regresión lineal del uso contra el tiempo: fecha estimada en que alcanza el total, con su intervalo (~95 %), crecimiento por día y bondad del ajuste	Sin pendiente positiva → «sin llenado previsible», sin fecha inventada. Un salto discreto (importar una base, un volcado) invalida la estimación hasta que la ventana lo absorbe: por eso se publican intervalo y ajuste, no solo la fecha
Certificado que expira	Aritmética sobre la fecha de vencimiento de los certificados observados	Fuera del umbral de aviso (30 días por defecto), silencio
Riesgo de RPO	Duración estimada del respaldo = tamaño ÷ velocidad **medida**; se proyecta cuándo alcanzará el límite (la ventana o el RPO objetivo)	Sin velocidad medida no se proyecta: error explícito
Tormenta de reinicios	Conteo en ventana deslizante (por defecto ≥ 5 reinicios en 1 hora por workload)	—
Ventanas de respaldo que se solapan	Intersección de las ventanas diarias declaradas (incluido el cruce de medianoche)	—

16.3 Recomendaciones de capacidad

Cada recomendación es **accionable pero nunca auto-aplicada**: genera una **revisión borrador de la receta** que un humano aprueba (y en workloads con datos, jamás se aplica sola). El propio objeto de

la recomendación nace con «requiere aprobación» y «no auto-aplicar» por construcción; el motor no tiene ningún método que aplique cambios.

- **Rightsizing:** compara el percentil 95 real de CPU/memoria contra lo declarado en la receta. Por debajo del 40 % de lo declarado → propone reducir; por encima del 85 % → propone ampliar; el valor propuesto es el p95 con un margen del 30 %, redondeado hacia arriba. Exige un mínimo de muestras (por defecto 100).
- **Consolidación:** nodos donde **todos** los workloads están infrautilizados, con las sumas de p95 frente a lo declarado encima de la mesa, para que la decisión sea revisable.
- **Ventanas:** convierte los solapes detectados en una propuesta de desplazamiento de ventana.

16.4 Panel de KPIs/KRIs de seguridad

Los insumos que la revisión por la dirección necesita, calculados desde las fuentes reales del sistema:

KPI/KRI	Definición	Sin datos
Edad de vulnerabilidades abiertas	Media en días desde que se vieron por primera vez, con desglose por severidad y la más antigua	0 abiertas **con datos** = 0 real; sin fuente, «sin datos»
Latencia de parcheo	Media descubrimiento → resolución de las resueltas, con percentil 90 y máxima	Sin resueltas → «sin datos»
Cobertura de recertificación	Recertificaciones **vencidas** completadas a tiempo ÷ vencidas. Las no vencidas no cuentan ni a favor ni en contra	Sin vencidas → «sin datos» (no un 100 % inventado)
Restores verificados	Puntos del catálogo verificados con restauración real ÷ total, por workload	Catálogo vacío → «sin datos»

Si una fuente falla o no registra nada, **solo ese** indicador sale como «sin datos» con su motivo; los demás se siguen mostrando. Que un KPI diga «sin datos» es en sí mismo la señal correcta para la revisión por la dirección: significa que nadie está alimentando esa fuente.

---

17. App móvil (PWA)

17.1 Qué es y qué no es

La superficie móvil es una **PWA endurecida** (aplicación web instalable), no una app nativa de tienda: llega antes, se actualiza con la consola y cubre lo que la guardia necesita. Su manifiesto es dinámico y white-label: el nombre y la marca de la app instalada salen de **su** instalación. Su alcance es **deliberadamente estrecho**: es el control remoto de guardia, **no** una segunda consola — no edita recetas ni configura políticas, y su lista de rutas es fija y comprobada.

Para instalarla: abra la dirección móvil de su consola (`/movil`) desde el navegador del teléfono e instálela desde el menú del navegador («Añadir a pantalla de inicio»). Requiere HTTPS.

17.2 Estado de un vistazo (solo lectura)

La portada móvil muestra: salud de la flota, **RPO actual** (antigüedad de la última copia correcta) y **RTO actual** (duración de la última ejecución terminada — una aproximación honesta hasta que el motor de DRP registre los tiempos reales de drill) y el estado de licencias. Pensada para el teléfono desde el que realmente se trabaja.

17.3 Alertas accionables

Con las notificaciones push activadas (llaves del servicio push generadas por su instalación), el teléfono recibe alertas con el **contexto mínimo para decidir**: caída de agente, drift en workload crítico, respaldo fallido y resultado de drill. Cada alerta enlaza a la portada móvil o a la aprobación correspondiente. Las suscripciones que el servicio push declara muertas se depuran solas.

> **Límite verificado — push:** el envío push de extremo a extremo requiere HTTPS, llaves configuradas y una suscripción de navegador real; está probado a nivel de unidades (la firma y el cifrado del mensaje) pero **no se probó contra un servicio push externo** en esta línea. Si activa las alertas, haga una **prueba real de recepción** en su teléfono antes de darlas por operativas.

17.4 Aprobaciones DRP desde el móvil

Desde **Aprobaciones** puede **aprobar, rechazar o vetar** (dentro de la ventana de aprobación diferida de §9.3) una ejecución de DRP con **la misma fuerza probatoria que en la consola**:

1. Abra la solicitud: verá el workload, el motivo y las ventanas (hasta cuándo no se ejecuta y hasta cuándo se puede vetar).
2. Elija la decisión. **Cada decisión exige su contraseña en la misma petición** (verificación por decisión, sin marcas de sesión reutilizables): es el step-up de la superficie móvil.
3. La decisión se escribe en el registro de auditoría con su **canal** (`movil`, igual que `consola` o `voz` cuando esos canales deciden) y se **encadena por hash** a la decisión anterior de su tenant: la cadena de decisiones es verificable.

Seguridad de la PWA: su código offline solo guarda la **carcasa estática pública**; nunca se cachean la API, las aprobaciones ni ninguna petición que no sea de lectura de la carcasa, y la navegación es «la red primero» con caída a la página de sin-conexión.

---

18. Solución de problemas y mensajes de error

18.1 Acceso y sesión

Síntoma / mensaje	Significado	Qué hacer
Error 404 al entrar	El identificador de tenant o la ruta no existen — o el recurso es de otro tenant (se responde igual a propósito)	Revise el identificador con quien le dio el acceso; no pruebe identificadores al azar

Síntoma / mensaje	Significado	Qué hacer
«Credenciales inválidas»	Correo o contraseña incorrectos en la plataforma o en el IdP local	Verifique y reintente. Tras varios fallos seguidos llega el bloqueo temporal: insistir solo lo prolonga
Error 429 (limitación de velocidad)	Demasiados intentos de login, de verificación o de decisión en poco tiempo	Espere el tiempo indicado y reintente una vez. Aplica a login, verificación pública de informes y decisiones móviles
La sesión se cierra sola	Expiró por duración máxima o por inactividad	Vuelva a entrar. Si ocurre trabajando, revise con su administrador los tiempos de su instalación
«Servicio de autenticación no disponible»	El proveedor de identidad no responde	En comercial, use su vía alternativa prevista si su instalación la tiene; en gobierno, el acceso local. Escale: sin IdP no hay login nuevo, pero los agentes y lo que corre no se detienen
La passkey no responde	Sin contexto seguro, navegador desactualizado o llave revocada/clonada	Entre por HTTPS con un navegador actual; o entre con contraseña y revoque/re-registre la llave en su seguridad
`step_up_required` (403)	Una acción sensible exige re-autenticación reforzada (§11.2)	Complete el step-up (contraseña y, si aplica, código) y reintente: el destino se conserva. La marca vale unos minutos
403 sin más detalle en una ruta	RBAC: su rol no contempla esa acción, o la ruta no tiene acción asignada (denegación por defecto)	No es un fallo de red: pida el rol correcto. La denegación queda registrada

18.2 Agentes y enrolamiento

Síntoma	Significado	Qué hacer
El token «no sirve» al enrolar	Venció (15 minutos), ya se usó (queda quemado) o es de otro tenant	Genere un token nuevo en la consola y repita el alta dentro de su vigencia
El agente arranca en «modo demo local»	No hay token en la configuración, la consola no responde o el endpoint de enrolamiento no existe	Revise `console_url` y el token en `/etc/abzu/agent.yaml` (permiso 0600) y la salida HTTPS del servidor; el agente lo dice en su registro
Error de certificado / la conexión se niega tras enrolar	La huella fijada de la consola no coincide (posible rotación no anunciada o está apuntando a otra consola)	Verifique que `console_url` es su consola. **No** ponga `insecure: true` en producción: resuelva el certificado

Síntoma	Significado	Qué hacer
Agente «sin conexión» en la consola	No llegan latidos: red, agente detenido o certificado revocado	En el servidor: estado del servicio y su registro. Recuerde: sus workloads siguen corriendo y el agente conserva el último estado deseado (§2.5)
Huella duplicada marcada para revisión	Dos agentes declaran la misma máquina (clonación o reinstalación)	Revise si es una reinstalación legítima y re-enrole con token nuevo; la marca existe para que un clon no duplique cupo en silencio
`status --json` con `agent_id` vacío	Ese agente está en modo demo local (nunca se enroló)	Enrólelo (§2.3) si debe contar como servidor gestionado
Sin contenedores en el inventario con el agente activo	No hay acceso al socket del motor en ese host	El agente lo reporta degradado con su motivo. Conceda el acceso al socket o acepte ese host sin gestión de contenedores

18.3 Recetas y catálogo

Síntoma	Significado	Qué hacer
La receta no guarda: error de validación con el campo	El YAML no cumple `abzu.recipe/v1` (p. ej. imagen sin etiqueta explícita, nombre inválido, un secreto en claro)	Corrija el campo indicado. Los secretos van solo como referencias `${secrets.<nombre>}`
La aplicación se bloquea «por admisión»	Una imagen quedó en cuarentena por vulnerabilidades según la política, o el escáner no está disponible y la política es fail-closed	Actualice la imagen a una versión sin el hallazgo o ajuste la política a sabiendas. No desactive el escaneo para «pasar»
Drift que vuelve una y otra vez	Algo fuera de la consola cambia el workload (o la receta no refleja una decisión ya tomada)	Use la **captura inversa** (§5.5): revise el borrador y promuévalo si el cambio es legítimo; si no lo es, la convergencia está haciendo su trabajo
Un servicio con datos no converge solo	Es deliberado: en `stateful` la política `manual` es forzosa	Apruebe la convergencia en la consola tras revisar el diff y tener snapshot previo
Entrada de catálogo «no verificada» / el agente rechaza la instalación	Firma ausente/inválida, publicador no confiable para ese agente, o el agente no tiene **ninguna** llave confiable configurada	Fije la huella del publicador por un canal independiente y configure sus llaves en los agentes. Sin firma válida no hay instalación: no hay excepción
La descarga del catálogo se bloquea con un motivo de compatibilidad	El destino, el motor o la versión de agente no cumplen lo declarado por la entrada	Instale en un destino compatible o pida al publicador una versión que declare su destino. La compatibilidad se aplica, no se salta

Síntoma	Significado	Qué hacer
La importación del bundle offline «no importa nada»	Falló alguna verificación (huella del emisor, firma del bundle o firma de alguna entrada) y la importación es atómica	Verifique la huella del emisor por el canal separado y que el bundle no se alteró en el transporte. No reintente con un bundle «parecido»

18.4 Respaldos, PITR y cinta

Síntoma	Significado	Qué hacer
El trabajo reporta «esperando ventana»	Está fuera de la ventana de respaldo declarada	Es el comportamiento correcto: espere a la ventana o dispare «Respaldar ahora» si es urgente (queda auditado)
La copia al segundo destino se aborta	La huella verificada durante la copia no coincide: la copia parcial se borra a propósito	Revise el destino y la red y reintente. Nunca fuerce a conservar una copia cuya huella no coincide
«El objeto está protegido» al intentar borrar un punto	El punto está sellado y/o con Object Lock vigente (GOVERNANCE/ COMPLIANCE)	Es la inmutabilidad funcionando. En GOVERNANCE solo un rol con permiso de bypass del almacenamiento podría — y no debe — hacerlo por fuera del producto. Espere al vencimiento de la retención
La verificación dice «escaneo omitido» (`skipped`)	El antivirus no está en el host del agente	El informe no finge: instale el escáner en ese host si su política exige el escaneo en cada verificación
La verificación falla el conteo	Las filas restauradas en sandbox no igualan **exactamente** el manifiesto	No use ese punto para producción hasta diagnosticar: puede ser un respaldo inconsistente o un sandbox con otra versión del motor. El RTO de ese punto no se da por medido
Restaurar un punto exige «aceptar el riesgo»	El punto está marcado **infectado** y su estado se propagó a las copias	Solo con aprobación explícita y registrada, y sabiendo que restaura contenido marcado. Prefiera siempre el último punto limpio verificado
PITR: «la cadena está rota» / no se puede restaurar a ese instante	Falta un eslabón (base o algún incremental intermedio) o la cadena no valida en orden	El sistema no promete un PITR imposible: elija un instante cubierto por una cadena completa y revise la retención — la poda nunca debería romper una cadena conservada; si lo hizo una acción manual fuera del producto, asúmalo en el diagnóstico
CDC: el RPO (lag) crece sin parar	El servicio de envío está caído o no puede subir segmentos: la posición solo avanza tras	Reviva el servicio de envío y verifique que sube y confirma segmentos. Mientras tanto, su RPO real es el lag que ve: no el de la política

Síntoma	Significado	Qué hacer
	subida verificada, así que el hueco es real	
La escritura a cinta falla «cartucho fuera del pool»	El cartucho cargado no pertenece al pool de la política	Monte un cartucho del pool. La lectura, en cambio, sigue al catálogo: para restaurar, monte el barcode que el catálogo indica
Reciclar un cartucho es rechazado	Tiene puntos con retención vigente — o el pool es WORM, donde ni forzando se puede antes de vencer	Espere al vencimiento. En WORM no hay vía de operador: es el sentido del pool
La consola avisa «se requiere un driver para tu equipo»	El equipo de cinta detectado exige driver propio según la tabla de equipos	Genere el requerimiento del agente (§8.3) y remítalo por el canal seguro. No instale paquetes de driver de otra procedencia: sin firma válida del proveedor, la instalación se rechaza entera
La instalación de un driver se rechaza	Sin llave pública configurada, firma inválida, huellas que no coinciden o equipo incompatible — en ese orden de comprobación	El intento queda registrado con su motivo. Corrija la causa (llave, paquete íntegro del canal seguro, equipo correcto); no reintente el mismo paquete alterado
Estado de cinta «sin hardware en este host»	No hay unidad/cambiador detectados en ese host (la detección también lo dice cuando las herramientas no están)	Ese host no puede ser servidor de cinta. La librería virtual sirve para pruebas y demos, no como cinta de producción

18.5 DRP, licencias y firmas

Síntoma	Significado	Qué hacer
La política autónoma «no actúa nunca»	Algún guardarraíl falla en cada evaluación: lo más común es que siga en borrador sin la **segunda firma**, que el nivel supere el techo de la instalación o que el kill switch esté activo	Abra la evaluación persistida: trae el resultado **por guardarraíl**. Active la política con la doble firma, ajuste el techo o desactive el interruptor, según corresponda
La evaluación queda en «rollback requerido»	La verificación posterior (salud o consistencia) falló **o no consta**	Trátela como una restauración no confiable: revise el workload restaurado y el drill asociado antes de cualquier nuevo intento autónomo sobre él
Nadie aprobó y «no se ejecutó nada con datos»		Es fail-closed funcionando: revise la cadena de guardia (números

Síntoma	Significado	Qué hacer
	La cadena de aprobación (consola → voz) no obtuvo un «autorizo» válido a tiempo	registrados, PIN) y vuelva a disparar por el canal que responda
La autorización por voz no se acepta	El webhook de la llamada no verificó su firma, o el PIN no coincide	No autorice DRP por voz hasta verificar la configuración de firma del webhook: es un requisito previo registrado, no un trámite
`entitlement_required` (403) en acciones de VMs	La función de pago de VMs está en estado «sin función» para su asignación	El inventario se sigue viendo: es bloqueo suave. Gestione la función en la asignación del proveedor (§10.4); en prueba, recuerde que todo funciona y el modal no bloquea
«No se pueden añadir servidores» con todo lo demás funcionando	Cupo agotado o prueba/suscripción vencida: **bloqueo suave**	Lo que corre no está en riesgo. Amplíe el cupo o renueve (§10.3); los servidores nuevos rechazados entran al liberarse cupo
Un servidor desaparecido «sigue contando» unos días	Su cupo se libera tras **7 días** sin reportarse	Espere a la gracia o verifique que el servidor realmente ya no debe contar y que no es un agente caído que volverá
La licencia offline «no valida»	Firma inválida, identificador de instalación distinto, versión/algoritmo no soportados o suscripción vencida	Pida el archivo emitido **para su identificador de instalación** y sin alterar. Cualquier edición del archivo invalida la firma
«No se puede firmar: sin identidad»	Su instalación no tiene identidad de firma activa cargada	Un administrador debe cargarla en **Configuraciones → Firma de informes** (§12.2) antes de la primera entrega
Un informe firmado «ya no consta como entregado»	El contenido cambió después de firmar: la comprobación verifica la firma contra el contenido **actual**	Vuelva a emitir y a firmar el informe correcto. Un informe sin firma registrada y verificada no cuenta como entregado (§12.3)
La evidencia exportada trae un control con la lista vacía	No hay hechos registrados para ese control en su instalación	Es la respuesta honesta del exportador: genere los hechos (opere, recertifique, verifique) o asuma el control como responsabilidad de su institución (§11.7)

18.6 Cómo pedir ayuda con evidencia

Cuando escale un problema, adjunte: (1) la ficha del agente o del workload con su foto de estado (`status --json` en el agente; incluye las secciones de cinta y de Fail2Ban con sus motivos), (2) el identificador del trabajo o de la evaluación en la consola, (3) las entradas de auditoría del intervalo (exportables) y (4) en respaldos, el **manifiesto** del punto afectado. Con eso, cualquier diagnóstico empieza en los hechos y no en suposiciones.

---

19. Buenas prácticas de operación

1. **Token de un solo uso por defecto;** llave de flota solo para ventanas de alta masiva, revocada al cerrarlas.
2. **`insecure: true` nunca en producción.** Si un enrolamiento «solo funciona así», el problema es el certificado: arréglelo.
3. **Todo lo que tenga datos, en `driftPolicy: manual` y con snapshot previo.** La convergencia automática es para lo que no tiene estado.
4. **Promueva la captura inversa el mismo día** del arreglo de emergencia: el borrador sin revisar es el drift de mañana.
5. **Escanee en la admisión y no lo desactive para «pasar»** un despliegue: la cuarentena es la función, no el obstáculo.
6. **3-2-1 desde el primer día:** segundo destino con credenciales propias, bucket dedicado con Object Lock y puntos críticos en COMPLIANCE. El repositorio endurecido no es opcional (§7.5).
7. **Verifique restaurando, no mirando:** política de verificación por workload + chequeo de integridad programado + drill mensual. El RTO que vale es el **medido**.
8. **Vigile el lag del CDC** como vigila el espacio en disco: un CDC caído en silencio es un RPO falso.
9. **El catálogo de cinta es dato crítico:** sin él no hay restauración posible desde cinta. Respáldelo fuera de la cinta.
10. **Políticas autónomas: empiece en `manual`, suba a `aprobación diferida` y solo entonces a `autónomo acotado`**, con drills bajo la política en cada nivel. El kill switch se prueba como se prueba un extintor: antes de necesitarlo.
11. **Doble firma de verdad:** el segundo firmante revisa alcance, ventana y RPO máximo; firmar sin leer convierte la SoD en un trámite y el evaluador no puede salvarle de eso.
12. **La asignación de licencias se toca solo en el panel del proveedor.** Si Abzu «no refleja» una función, diagnostique la validación y su gracia antes de buscar un registro paralelo: no existe, a propósito.
13. **Configure la identidad de firma antes de la primera entrega** y trate su llave privada como la llave que es: cifrada en reposo, nunca exportada por pantalla.
14. **Alertas que alguien atiende:** suscríbase por severidad y por canal con criterio. Revise trimestralmente qué alertas llegaron y quién actuó.
15. **Recertifique en plazo:** la cobertura de recertificación es un KPI de la revisión por la dirección (§16.4); dejarla vencer se ve, y debe verse.

16. **En gobierno, telemetría cero por defecto:** ninguna salida de la instalación sin autorización explícita y registrada de la institución.

---

20. Glosario

- **Agente:** programa ligero instalado en cada servidor o clúster gestionado; inicia siempre la conexión hacia la consola (phone-home) y se autentica con certificado propio (mTLS).
- **Receta:** documento YAML `abzu.recipe/v1` que declara el estado deseado de una aplicación. Es la fuente de verdad; lo demás se genera desde ella.
- **Revisión:** versión inmutable de una receta; cada cambio crea una nueva (un commit) y nunca reescribe las anteriores.
- **Drift (desviación):** diferencia entre el estado real y el deseado. Su tratamiento depende de la política: `auto`, `manual` o `report`.
- **Captura inversa:** generar una revisión borrador de la receta desde el estado real, para aprobación humana.
- **Manifiesto (de respaldo):** documento JSON que acompaña a cada punto de restauración con su huella, conteos y tiempos medidos.
- **PITR:** restauración a un instante concreto dentro de la ventana de retención, reproduciendo la cadena base + incrementales hasta ese momento.
- **CDC:** respaldo continuo: envío de segmentos de log (WAL/binlog) a medida que se cierran, con confirmación solo tras subida verificada. Su métrica es el lag.
- **Punto sellado:** punto GFS (semanal/mensual/anual) que no se puede podar ni, con Object Lock, borrar hasta que vence su periodo.
- **Sandbox (de verificación):** entorno aislado y desechable, sin red hacia producción, donde se restaura para verificar.
- **Runbook:** receta especial que ordena la reconstrucción ante un desastre (red → volúmenes → datos → aplicaciones).
- **Guardarraíl:** límite verificable de una política autónoma (alcance, ventana, RPO, snapshot previo, techo, kill switch). Si uno falla, no hay acción autónoma.
- **Kill switch:** interruptor global por instalación que anula toda autonomía del DRP al instante.
- **Seat:** unidad de licencia: un agente enrolado. Los contenedores por agente son ilimitados.
- **Bloqueo suave:** ante un problema de licencia o de función de pago, se detiene lo nuevo y lo existente sigue intacto.
- **Step-up:** re-autenticación reforzada ante un contexto anómalo, exigida antes de acciones sensibles.
- **Bundle:** paquete firmado y transportable sin red (instalación gobierno, catálogo offline, licencia offline). Se verifica por firma antes de usarse; si no verifica, no se usa.
- **Fail-closed:** ante la duda — firma, licencia, política, canal — el sistema no actúa y escala. Es el comportamiento por defecto de todo el producto.

Fin del manual. SEGRD — Manual de usuario de Abzu v2, 2026-10-18. Este documento describe el producto según su código y diseño vigentes en la rama integrada `main`; los límites verificados se declaran en el apartado correspondiente de cada función.