

Documentación de OpenLatch

Instala el cliente, define la Autonomy Zone de cada agente y despliega en toda tu flota.

Instalar el cliente

Cómo funcionan los veredictos →

Primeros pasos



Instala el cliente y obtén tu primer veredicto.

Conceptos



Autonomy Zones, políticas de intención y veredictos.

Guías



Una tarea por página, de la redacción a Enforce.

Desplegar



Despliega el cliente con MDM o gestión de configuración.

Referencia



Cada comando, código de error y campo de política.

Registro de cambios



Cada versión del cliente y cada cambio de la plataforma.

Introducción

Este recorrido te lleva de una máquina recién instalada a tu primera acción decidida: instalas el cliente junto a un agente, inicias sesión en la consola, escribes una política de intención y la ves decidir.

OpenLatch es la capa de control de ejecución para agentes de IA empresariales. El cliente se ejecuta junto a cada agente y decide sobre cada acción cubierta antes de que se ejecute. La plataforma, en `app.openlatch.ai`, es donde defines la Autonomy Zone de cada agente y consultas el registro de lo que ocurrió.

- 1

[Instala el cliente](#) en una máquina donde se ejecute un agente de código.
- 2

[Inicia sesión en la consola](#) con el método que usa tu organización.
- 3

[Escribe tu primera política](#) y ratifícala en Monitor.
- 4

Usa el agente como siempre y luego abre **Anomalies** en la consola para ver lo que registró la política.

Quién hace qué

Eres	Empieza por
Desarrollador con un agente de código	Instalar el cliente
Administrador que despliega el cliente en muchas máquinas	Instalar el cliente y luego la sección Desplegar
Responsable de seguridad o cumplimiento que escribe políticas	Escribir tu primera política y luego Políticas de intención

Antes de empezar

- Una máquina macOS, Linux o Windows con al menos un agente de código que el cliente conecta. [Niveles de control](#) los enumera, plano por plano.
- Una cuenta en una organización de OpenLatch en `app.openlatch.ai`.

Para entender qué es OpenLatch antes de instalar nada, lee [Qué es OpenLatch](#).

Instalar el cliente

Instala el cliente en la máquina donde se ejecuta tu agente de código, para que cada acción cubierta se decida antes de ejecutarse. El cliente se instala en el ámbito del usuario y no necesita derechos de administrador. Elige el comando que corresponda a la máquina.

En cualquier máquina que ya tenga Node.js:

```
npx @openlatch/client init
```

Solo npx necesita Node.js. El cliente en sí es un binario nativo y no se ejecuta sobre Node.js.

No requiere ningún runtime:

```
curl -Lsf https://openlatch.ai/install.sh | sh
```

No requiere ningún runtime:

```
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
```

Cada comando instala la misma versión del cliente desde el registro de npm y luego ejecuta `openlatch init`, que detecta los agentes de código de la máquina, conecta sus hooks e inicia el cliente.

Registrar la máquina

- 1 En una máquina donde estás frente a una terminal, `openlatch init` abre `app.openlatch.ai` en tu navegador cuando la máquina aún no tiene credencial. Inicia sesión y el cliente guarda la credencial en el llavero del sistema operativo.
- 2 En una máquina sin nadie frente a la terminal, crea primero una clave de API en la página **API Keys** de tu cuenta en la consola y defínela en `OPENLATCH_API_KEY` antes de ejecutar el comando de

instalación.

- 3 Ejecuta `openlatch status` y comprueba que cada agente aparece con su nivel de control.

Nota

La descarga se verifica con el resumen publicado por el propio registro. Eso detecta corrupción y truncamiento, no un registro comprometido.

Comprobar los hooks

Ejecuta `openlatch doctor`. Diagnostica la configuración y la conexión con la plataforma y, para todo lo que no está bien, indica la causa y el comando que lo corrige.

Qué cambia la instalación

- Instala solo para tu usuario. No se necesitan derechos de administrador.
- Con un script, coloca dos binarios, `openlatch` y `openlatch-hook`, en `~/local/bin` en macOS y Linux, o en `%LOCALAPPDATA%\openlatch\bin` en Windows, y añade esa carpeta a tu PATH. Puede que una terminal ya abierta tenga que reabrirse una vez.
- Con `npx`, no se añade nada a tu PATH: el cliente guarda sus binarios en su directorio de estado, así que puede que no se encuentre un `openlatch` sin más. Ejecuta los comandos siguientes con el mismo comando `npx`, con el comando que quieras en lugar de `init`, por ejemplo `status`.
- Antes de instalar nada, compara un resumen SHA-512 de la descarga con el que el registro publica para esa versión.

Código de salida

Para las herramientas de despliegue que leen el código de salida de un script de instalación:

Código	Significado
0	Instalado y configurado. El resultado aún puede señalar algo que revisar
6	Instalado, pero la configuración no terminó. El resultado en pantalla dice por qué
7	Instalado, a la espera de un agente de IA: la máquina aún no tiene ninguno. Instala uno y vuelve a ejecutar el script
2-5	No se instaló nada utilizable. El script muestra un código 0L-21xx que indica el motivo

Plataformas compatibles

Plataforma	Estado
macOS, Apple silicon e Intel	Compatible
Linux con glibc, x86_64 y arm64	Compatible
Windows, x64	Compatible
Linux con musl, como Alpine	No compatible. El script de instalación se detiene e indica el motivo
Windows en ARM64	No compatible. El script de instalación se detiene e indica el motivo

A continuación, [escribe tu primera política](#).

Escribir tu primera política

En este tutorial conviertes una frase en una Agent Intent Policy (AIP), la ratificas en Monitor y ves cómo registra una anomalía sin detener al agente.

Antes de empezar, [instala el cliente](#) en una máquina con un agente de código e [inicia sesión en la consola](#).

Escribir la política

1 En la consola, abre **Autonomy Zone**, luego la lista de políticas, y empieza a crear una nueva política de intención.

2 En el paso **Brief**, pega una frase que diga lo que quieres o elige un ejemplo. Este tutorial usa:

```
Don't let the agent fetch anything from pastebin.com.
```

3 En el paso **Policy**, lee la frase que OpenLatch generó a partir de tu brief, con los casos de prueba mostrados al lado. Cada caso es una situación concreta y el veredicto que devolvería la política. Responde Sí si la frase dice lo que querías decir. Responde No y di con tus propias palabras lo que querías decir: OpenLatch regenera la frase y sus casos.

4 En el paso **How it runs**, elige el modo **Monitor**.

5 En el paso **Ratify**, elige la zona que contiene la política y después ejecuta la reproducción. Aplica la política a la actividad reciente de tus agentes y muestra lo que habría decidido. Ratifica la política.

app.openlatch.ai/zone/policies/new

OpenLatch

Autonomy Zone / New intent policy

Q Ctrl+K

?

99+ 99+ DC

Crispy Pay

MONITOR

- Home
- Agents
- Anomalies
- Inventory
- Activity Intelligence

MANAGE

- Autonomy Zone
- Activity Digests

SYSTEM

- Audit Trail
- Settings

New intent policy

Turn what you already have into an intent policy. Nothing runs until you ratify it.

1 Brief 2 Policy 3 How it runs 4 Ratify

Define the policy intent

Provide a brief or reference material. OpenLatch will use it to draft an Agent Intent Policy for your review.

☐ Paste a brief
Type or paste what you mean, in your own words.

☐ Upload a policy
A document, an SOP, a procedure. We read it and pull the intents out.

☐ Record a conversation
Describe the rule out loud, or attach the meeting where it was agreed.

Your brief Examples ▾

e.g. Payments engineering change-control SOP:

- No production deployment or database migration without an approved change ticket.
- During declared incidents, the on-call engineer may act without a ticket, recording actions on the incident bridge.
- Destructive commands (DROP TABLE, rm -rf, kubectl delete namespace) are strictly forbidden on production clusters.
- All access to customer secrets or payment credentials must require an active session.

One rule per line. Anything we infer is marked as inferred, on the line itself.

Write what the agent does, not what it intends

- Name the exact action: command, path, domain or tool (e.g. "Block Bash when command.program equals rm").
- Say the outcome: block it, or ask a human before it runs.
- Phrase exceptions as a fact the agent shows: "unless agent.environment equals development".

This policy will be created in Crispy Pay. [Turn it into a policy →](#)

El asistente para crear una política de intención: Brief, Policy, How it runs, Ratify.

Si el asistente indica que la redacción no está habilitada, un administrador de tu sitio de OpenLatch todavía no ha configurado la redacción asistida por IA.

Verla registrar

- 1 La ratificación reconstruye el bundle de políticas. El cliente lo obtiene en su siguiente consulta; para obtenerlo ya, ejecuta `openlatch system aip refresh` en la máquina del agente.
- 2 Pide al agente que obtenga una página de pastebin.com.
- 3 La política se ejecuta en Monitor, así que la acción continúa y se registra una anomalía. Abre **Anomalies** en la consola para leerla: la anomalía indica la política, su dimensión y la zona de la que proviene.

Lo que hiciste

Escribiste una política de intención con tus propias palabras, la comprobaste con casos concretos y la ejecutaste en Monitor, donde registra sin detener nunca al agente. Cuando las anomalías muestren que la política es correcta, pásala a Enforce en su zona: el cliente aplica entonces su veredicto antes de que se ejecute la acción. [Modos](#) explica la diferencia, y [Políticas de intención](#) explica lo que OpenLatch construyó a partir de tu frase.

Iniciar sesión en la consola

Inicia sesión en la consola en `app.openlatch.ai` para definir Autonomy Zones, escribir políticas de intención y consultar lo que hicieron tus agentes.

Iniciar sesión

- 1 Abre `app.openlatch.ai`. La página de inicio de sesión ofrece los métodos que permite tu organización.
- 2 Inicia sesión con uno de ellos: el inicio de sesión único SAML de tu organización, una cuenta de Google, Microsoft, GitHub, Apple o LinkedIn, un enlace mágico enviado por correo electrónico, una clave de acceso, o una dirección de correo y una contraseña.
- 3 Si tu cuenta tiene activada la autenticación en dos pasos, introduce el código de tu aplicación de autenticación.

La primera vez que inicias sesión sin invitación, la consola te guía para crear tu organización. Si un compañero te invitó, el enlace de invitación te lleva directamente a tu organización.

Vincular el cliente a tu cuenta

Cuando `openlatch init` u `openlatch system auth login` necesita una credencial, abre la consola en tu navegador. Una vez iniciada la sesión, la consola entrega a la terminal un código de un solo uso, nunca una clave. El cliente lista tus organizaciones con ese código y solo entonces crea una clave de API, en la organización elegida, y la guarda en el llavero del sistema operativo.

Tus organizaciones	Qué ocurre
Una	Se elige por ti, y la terminal muestra su nombre
Varias, en una terminal	La terminal pregunta cuál. <code>--org <ORG></code> u <code>OPENLATCH_ORG</code> la nombra de antemano, por su slug, su nombre o su identificador
Varias, sin nadie a quien preguntar (<code>--yes</code> , <code>--json</code> , <code>--quiet</code> o sin terminal)	<code>openlatch system auth login</code> se detiene con <code>OL-1608</code> y las lista. <code>openlatch init</code> sin clave de API nunca inicia una sesión en el navegador, así que se detiene antes, con <code>OL-1600</code> : define <code>OPENLATCH_API_KEY</code> o ejecuta <code>init</code> en una terminal

Una máquina permanece en la organización en la que se inscribió. `openlatch system auth switch --org <ORG>` la mueve a otra de tus organizaciones.

Las páginas de inicio de sesión

Página	Para qué sirve
/login	Iniciar sesión con cualquier método que permita tu organización
/register · /signup	Crear una cuenta, desde una invitación o por tu cuenta; /signup lleva a /register
/forgot- password	Pedir por correo un enlace para restablecer la contraseña
/reset- password	Elegir una nueva contraseña desde el enlace de restablecimiento
/two-factor- verify	Introducir el código de tu aplicación de autenticación
/change- password	Elegir una nueva contraseña cuando tu cuenta debe cambiarla antes de continuar
/welcome	Crear tu organización después de tu primer inicio de sesión
/cli-auth	La página que abre el cliente para recibir su inicio de sesión; la organización se elige en la terminal
/activate	Una pantalla de configuración única que muestra un sitio nuevo de OpenLatch antes de que exista su primera organización. En <code>app.openlatch.ai</code> nunca la verás

Qué es OpenLatch

Lee esta página para entender qué hace OpenLatch antes de instalarlo.

OpenLatch es la capa de control de ejecución para agentes de IA empresariales. Te permite dar más autonomía a los agentes sin perder el control: defines el terreno de juego de cada agente, y cada acción cubierta que el agente realiza se decide antes de ejecutarse.

El problema

Los agentes de IA hacen más que responder. Ejecutan comandos, modifican archivos, llaman a herramientas y gastan dinero, a menudo con permisos amplios en la máquina donde se ejecutan. Las instrucciones de un prompt orientan lo que un agente pretende hacer, pero no pueden garantizar lo que hace: un contexto largo, un plan confuso o una instrucción inyectada todavía pueden llevar a una acción que nadie quería.

El enfoque

Principio	Qué significa
Dentro del agente, no un proxy delante de él	El cliente se conecta a los hooks nativos del propio agente, así que ve cada acción mientras aún hay tiempo de cambiar el resultado
Decidido antes del impacto	Cada acción cubierta llega al punto de decisión antes de ejecutarse, y el veredicto se aplica. Un veredicto es determinista porque lo produjo una política explícita
Autoridad local	El cliente decide en la máquina, a partir de un bundle de políticas residente. Sigue decidiendo con la red desconectada, y el agente nunca depende de la red para un veredicto
Tu intención, con tus palabras	Escribes una política como una frase. OpenLatch la convierte en una Agent Intent Policy y te muestra casos de prueba concretos antes de que la ratifiques
Evidencia a posteriori	El cliente envía a la plataforma el registro de cada acción y su veredicto después de la decisión, para la auditoría y la revisión

El modelo de amenaza es un agente que se comporta mal, no un operador malintencionado: OpenLatch controla lo que hace un agente autorizado a través de sus propios hooks. No es un sandbox imposible de eludir.

Las piezas

Pieza	Dónde se ejecuta	Qué hace
El cliente	Junto a cada agente, en un equipo de desarrollo o dondequiera que se ejecute el agente	Conecta los hooks del agente, guarda el bundle de políticas, decide cada acción cubierta y la registra
La plataforma	<code>app.openlatch.ai</code>	Guarda el registro de agentes, las Autonomy Zones y sus políticas de intención, compila y distribuye los bundles de políticas, y conserva la evidencia

Próximos pasos

- [Instala el cliente.](#)
- Lee [Cómo funciona OpenLatch](#) para seguir una acción de principio a fin.
- Consulta qué agentes están conectados, plano por plano, en [Niveles de control](#).

Preguntas frecuentes

Aquí encontrarás respuestas rápidas, cada una con un enlace a la página que trata el tema por completo.

¿OpenLatch es gratuito?

El precio de la plataforma se está definiendo con nuestros socios de diseño, y los pilotos con esos socios son gratuitos.

¿Necesito Node.js?

Solo para `npx @openlatch/client init`, porque `npx` forma parte de Node.js. El cliente es un binario nativo y no se ejecuta sobre Node.js. En una máquina sin Node.js, ejecuta el script de tu plataforma; no necesita nada más que lo que trae el sistema operativo:

```
curl -LsSf https://openlatch.ai/install.sh | sh
```

```
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
```

Consulta [Instalar el cliente](#).

¿Qué agentes cubre OpenLatch?

Claude Code, Codex CLI, Cursor y Cline están conectados hoy. La cobertura se indica por agente y por plano: [Niveles de control](#) muestra qué plano es Enforced, Monitored u Observed para cada uno.

¿Qué datos envía el cliente a la plataforma?

Después de cada decisión, el cliente envía el registro de la acción y su veredicto: lo que el agente intentó hacer, como un comando, una ruta de archivo o un destino de red, y lo que se decidió. Las credenciales se ocultan en la máquina antes de que nada salga de ella. Consulta [Privacidad y telemetría](#).

¿Puedo usar OpenLatch sin conexión?

Sí. El cliente decide en la máquina con su bundle de políticas residente, que sigue decidiendo con la red desconectada. Obtener políticas nuevas y enviar registros a la plataforma necesitan conexión; decidir, no.

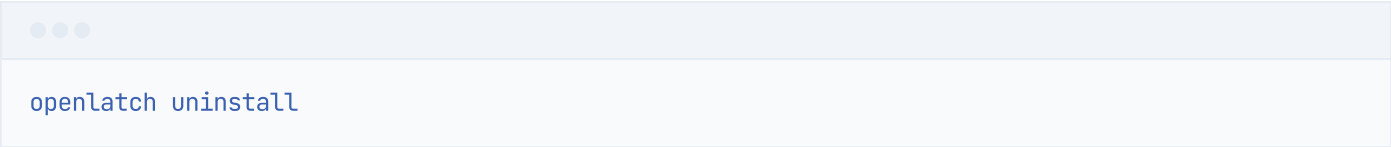
¿OpenLatch ralentiza mi agente?

El cliente decide en la máquina, a partir de un bundle de políticas que guarda en memoria, así que el agente nunca depende de la red para un veredicto. Los registros se envían a la plataforma después.

¿Puedo alojar la plataforma yo mismo?

Hoy no. La plataforma funciona como servicio gestionado en `app.openlatch.ai`.

¿Cómo desinstalo el cliente?

A terminal window with a light blue header bar containing three dots. The main area is white and contains the command `openlatch uninstall` in blue text.

```
openlatch uninstall
```

Esto quita los hooks de los agentes y detiene el cliente. Añade `--purge` para borrar también el estado del cliente, incluida la carpeta `~/ .openlatch` y la credencial guardada en tu llavero.

Ninguno de los dos comandos elimina los binarios: una purga no desinstala los archivos que colocó la instalación. Si instalaste con un script, `openlatch` y `openlatch-hook` siguen en `~/ .local/bin` o `%LOCALAPPDATA%\openlatch\bin`, igual que la entrada del PATH. Bórralos a mano si quieres la máquina exactamente como estaba.

¿Dónde informo de un error o pido una función?

Escríbenos a hello@openlatch.ai.

Conceptos

Estas páginas explican las ideas que encontrarás en todo OpenLatch: la Autonomy Zone, las políticas de intención, los veredictos, los modos y las anomalías. Léelas para entender por qué el producto se comporta como lo hace; las guías te dicen dónde hacer clic.

OpenLatch ejecuta un bucle de control continuo en torno a cada agente. El control existe para que sea más fácil decir que sí: cuanto más claro es el terreno de juego de un agente, más se le puede confiar que haga por su cuenta.

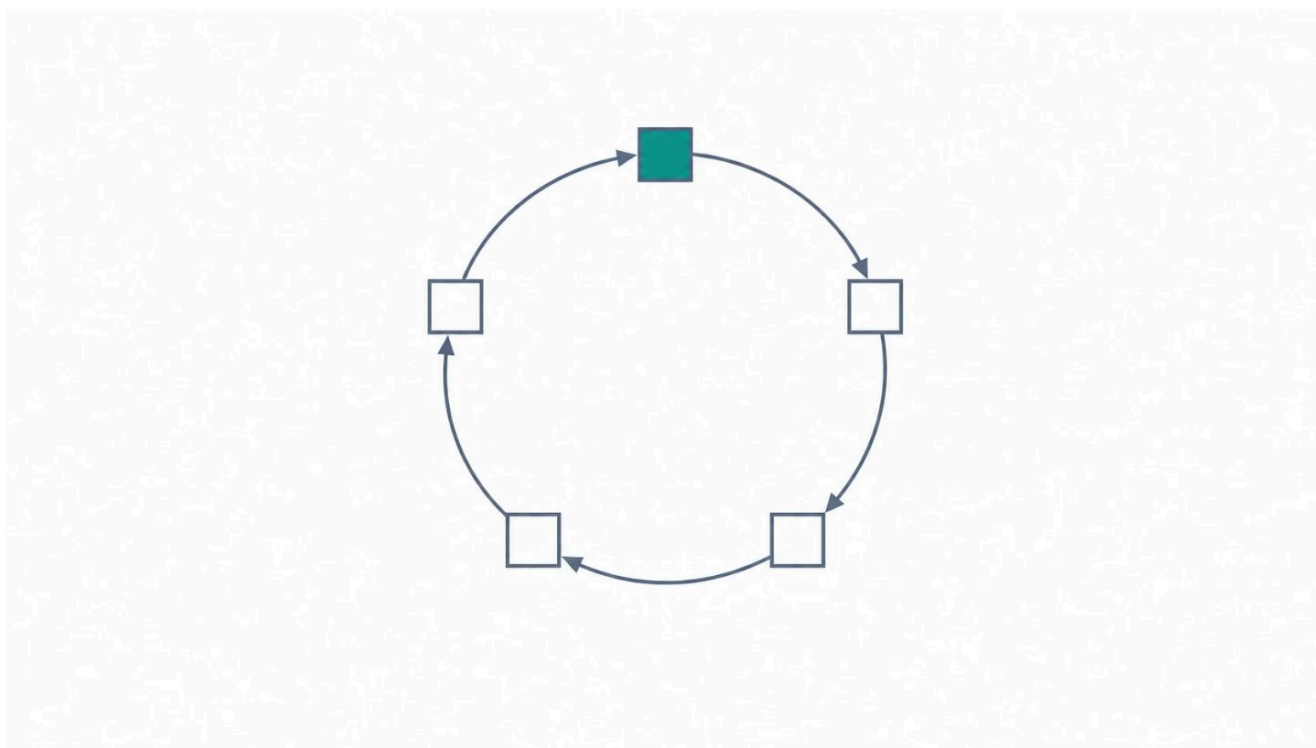


Figura 1. El bucle de control: Identificar, Definir, Detectar, Actuar y Mejorar, en torno a cada agente.

Paso	Qué ocurre	Para leer
Identificar	El agente se conecta al contexto de tu empresa: su identidad, su propietario, su unidad, su tipo y su entorno, y el nivel de control que OpenLatch tiene sobre él	Niveles de control
Definir	Cada agente se resuelve en una Autonomy Zone efectiva, construida con políticas de intención en cinco dimensiones	Autonomy Zone · Dimensiones · Políticas de intención
Detectar	Lo que hace el agente se captura como un grafo de ejecución y se contrasta con su zona. Una acción que infringe una política de intención es una anomalía	Grafo de ejecución · Anomalías
Actuar	Antes del impacto, cada acción cubierta recibe un veredicto: Allow, Ask, Block u Optimize	Veredictos · Modos
Mejorar	El registro de cada decisión muestra qué políticas son correctas, para que puedas pasarlas de Monitor a Enforce. OpenLatch nunca amplía por su cuenta la autoridad de un agente	Modos

Para seguir una acción desde el hook del agente hasta el veredicto, lee [Cómo funciona OpenLatch](#).

Autonomy Zone

La Autonomy Zone de un agente es el perímetro operativo contextual en el que actúa: lo que intenta lograr, las condiciones en las que puede actuar y las consecuencias que la empresa acepta. Dentro de la zona, el agente es libre de perseguir su misión. Cada acción cubierta que cruzaría el límite se decide antes del impacto.



Figura 1. Una Autonomy Zone es el perímetro operativo de un agente: lo que puede hacer, en qué condiciones y con qué consecuencias aceptadas.

El agente aporta la misión; la zona aporta las reglas. Piensa en una cancha de baloncesto: las líneas y las reglas del juego son fijas, el jugador se mueve libremente dentro de ellas y el árbitro pita lo que cruza una línea. La zona es la cancha, sus políticas de intención son las reglas y el veredicto es la decisión del árbitro.

Qué contiene una zona

Una zona contiene políticas de intención en cinco dimensiones: Calidad, Economía, Seguridad, Protección y Cumplimiento. [Dimensiones](#) explica qué cubre cada una. Cada política se ejecuta en modo Disabled, Monitor o Enforce en esa zona, como explica [Modos](#).

Las zonas siguen a tu organización

Las zonas se colocan sobre el árbol de unidades de tu organización, normalmente sincronizado desde tu directorio. No mantienes una segunda jerarquía.

Regla	Qué significa
Una zona vive en una unidad	Cualquier unidad puede tener una zona, y la mayoría no la tiene
La herencia nunca es opcional	Una unidad sin zona hereda la zona de la unidad más cercana por encima de ella que tenga una. Cada zona hereda de todas las zonas por encima de ella
Añadir, nunca relajar	Una zona inferior puede añadir políticas, nunca quitar lo que hereda
Un agente, una zona	Cada agente se asigna exactamente a una zona: por defecto, la zona más cercana en la unidad de su propietario o por encima, o la que un usuario fija a mano
Sin selección oculta	Una política no puede aplicarse a unos agentes de su zona y no a otros. Si no debe alcanzar a un agente, mueve el agente o escribe la política más abajo en el árbol

La zona efectiva

Aquello bajo lo que opera realmente un agente es su **zona efectiva**: todas las políticas de la zona a la que está asignado y de cada zona por encima, agrupadas por dimensión. Cada política indica la zona de la que proviene. Cuando la misma política está en varios niveles del camino de un agente, le llega una sola vez, en el más estricto de sus modos.

OpenLatch

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / Autonomy Zone

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1828.171.37.168IDLE

Autonomy Zone

12 intent policies reach this agent from 2 Autonomy Zone layers – none authored on the agent.

AUTONOMY ZONE

Payments

inherits 2 layers: Crispy Pay

POLICIES IN FORCE

12

9 inherited · 3 in its Autonomy Zo...

ENFORCING

6 of 12

the other 6 record, and do not act

ANOMALIES · 14 DAYS

0

nothing flagged

2 Autonomy Zone layers apply here

Assigned to Payments, resolved top-down from the org tree. A layer below may tighten what it inherits.

Crispy Pay

+9151

Payments ASSIGNED HERE

+344

Effective intent policies

INTENT POLICY	ALSO IN	LAYER	MODE	ANOMALIES 14D
▼ Economics1 policy · 0 enforcing · 1 in Monitor				
AIP-007 A session stays under sixty tool runs	–	↑ Crispy Pay	MONITOR	0

La pestaña Autonomy Zone de un agente: su zona y su camino, las políticas en vigor y la zona de la que proviene cada una.

Qué no es una zona

No es	Porque
Una lista fija de salvaguardas	La misma acción puede recibir otro veredicto para otro agente, otra misión u otro entorno
Instrucciones en el prompt	Un prompt orienta el comportamiento y puede olvidarse. La zona se aplica en el límite de ejecución del agente, fuera de su ventana de contexto
Una lista de permisos	Las identidades y permisos existentes alimentan la zona, que juzga la acción en contexto
Un sandbox imposible de eludir	El modelo de amenaza es un agente que se comporta mal, no un operador malintencionado

Dimensiones

Una Autonomy Zone está delimitada por cinco dimensiones. Cada una responde a una pregunta sobre la acción de un agente, y cada una se llena de políticas de intención.

Dimensión	La pregunta que responde	Ejemplos de políticas de intención
Calidad	¿El trabajo está a la altura?	El código se prueba antes de publicarse. Los resultados cumplen el nivel de precisión y exhaustividad exigido. Se sigue el proceso establecido
Economía	¿Está justificado el coste?	Presupuestos y topes de tokens por periodo. Sin bucles descontrolados ni reintentos inútiles. El modelo se ajusta a la tarea
Seguridad	¿Puede el agente tocar esto?	Ninguna credencial sale del perímetro corporativo. Solo herramientas, servidores MCP y skills aprobados. El acceso a datos se mantiene dentro de lo autorizado
Protección	¿Es aceptable la consecuencia?	Ningún acceso a producción sin aprobación. Ninguna operación irreversible o destructiva. El radio de impacto y el impacto en clientes se mantienen dentro de lo que la empresa acepta
Cumplimiento	¿Es legal y conforme a las normas?	Restricciones de protección de datos para los agentes que atienden a clientes en mercados regulados. Políticas y normas internas. Obligaciones contractuales

Una política puede abarcar varias dimensiones

Una política de intención se compone de uno o varios átomos de intención, y cada átomo pertenece a una dimensión. Una política lleva las dimensiones de todos sus átomos, así que una misma política puede tratar a la vez de Seguridad y de Protección. La consola agrupa las políticas efectivas de un agente por dimensión, y cada anomalía indica la dimensión de la política que infringió.

Lo que no es una dimensión

- **La misión.** Para qué sirve un agente pertenece al agente, no a la zona. El agente aporta la misión; la zona aporta las reglas.
- **La evidencia de auditoría.** Las reglas de cumplimiento se ejecutan en la zona, antes de la acción. La evidencia de lo que ocurrió, de qué política se aplicó, de qué veredicto se devolvió y de si se aplicó es el registro que OpenLatch conserva de cada decisión, sea cual sea la dimensión.

Políticas de intención

Una política de intención expresa lo que quieres decir, en lugar de cada superficie que un agente podría usar para hacerlo. En OpenLatch la escribes como una **Agent Intent Policy (AIP)**, y OpenLatch la compila en reglas deterministas que el cliente aplica antes de cada acción cubierta.

De tus palabras a una AIP

Partes de un brief: una frase, un documento de políticas pegado o una conversación grabada. OpenLatch lo divide en **átomos de intención**, uno por regla, y genera dos cosas a partir de cada átomo: una frase legible y casos de prueba concretos.

Campo del átomo	Qué contiene	Ejemplo
who	Los agentes a los que se refiere la regla	cualquier agente
does what	La acción	desplegar
to what	El objetivo	un namespace de producción
only if	La condición que lo permite	hay un ticket de cambio aprobado vinculado
unless	La excepción	hay una sala de crisis de incidente activa
class	Si el átomo se puede aplicar: aplicable, solo observable o fuera de alcance	aplicable

Nunca rellenas estos campos. OpenLatch los extrae de tu brief, y tú los cambias editando la frase.

Ratificas la frase

Para el átomo anterior, lees esta frase:

Mientras no haya una sala de crisis de incidente activa, cuando un agente solicite un despliegue en producción, el sistema exigirá un ticket de cambio aprobado.

Al lado, los casos de prueba muestran lo que decidiría la política en situaciones concretas, por ejemplo un despliegue con ticket, otro sin ticket y otro sin ticket durante un incidente. Los casos están ahí para que tu respuesta sea informada. Respondes Sí o No sobre la frase:

- **Sí** acepta el átomo.
- **No** abre una nota donde dices con tus propias palabras lo que querías decir. OpenLatch cambia los campos, regenera la frase y sus casos, y marca los casos cuyo veredicto cambió. El átomo vuelve para un nuevo Sí o No, porque una respuesta dada sobre la frase anterior no cubre la nueva.

Nunca ves ni apruebas las reglas compiladas. Apruebas lo que puedes leer.

De una AIP al cliente

Una vez aceptados sus átomos, ratificas la AIP en cero, una o varias zonas, cada una en su propio modo. La plataforma compila los átomos en reglas y construye un bundle de políticas por agente a partir de su zona efectiva. Se verifica la integridad del bundle, y el cliente lo guarda en la máquina, así que sigue decidiendo con la red desconectada.

La clase de un átomo cuenta en ejecución: un átomo `monitor-only` se queda en Monitor incluso cuando su política se ejecuta en Enforce, porque se puede observar pero no aplicar.

El ciclo de vida de una política

1. Escribe un brief o aporta un documento de origen.
2. Revisa las frases generadas y sus casos, y acéptalas.
3. Reproduce la política sobre la actividad reciente de tus agentes para ver lo que habría decidido.
4. Ratifícala en Disabled, Monitor o Enforce.
5. Ejecútala en Monitor, lee sus anomalías y pásala a Enforce cuando sea correcta.

[Escribir tu primera política](#) recorre estos pasos en la consola.

Veredictos

Antes de que se ejecute una acción cubierta, el cliente devuelve uno de cuatro veredictos: Allow, Ask, Block u Optimize. Un veredicto es determinista porque lo produjo una política explícita.

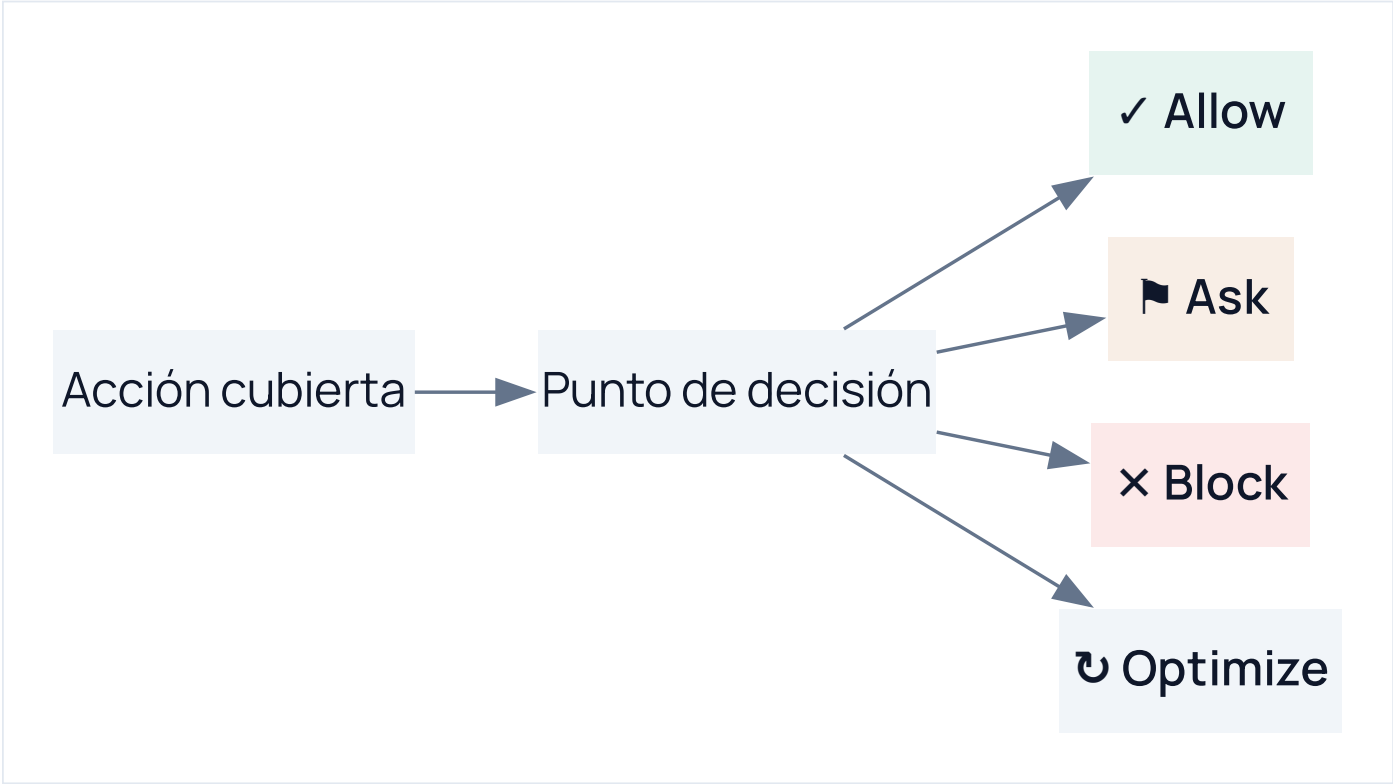


Figura 1. Cada acción cubierta llega al punto de decisión antes de ejecutarse. Las políticas de intención de la zona efectiva producen un veredicto determinista.

Veredicto	Qué ocurre
Allow	La acción se ejecuta. En Monitor, puede ejecutarse y quedar marcada
Ask	Por defecto, la acción se ejecuta y se marca para una revisión asíncrona; solo se retiene cuando un átomo lo activa para una acción irreversible
Block	La acción no se ejecuta. El agente recibe un rechazo nativo que nombra la regla que lo produjo
Optimize	El camino cambia. En Enforce, una palanca actúa; en Monitor, mide lo que habría hecho

Cuando varias políticas coinciden con una misma acción, gana el veredicto más restrictivo, en este orden: Block, después Ask, después Optimize, después Allow.

Ask

Ask deja que la acción se ejecute y la marca para revisión, para que una persona decida si el agente debe volver a hacerlo. Evita la repetición, no la primera vez. Un átomo puede optar por retener una acción irreversible hasta que se responda; nada más depende de una persona.

Cómo llega Ask al agente depende del agente. Claude Code lo recibe como su propia solicitud nativa, con el motivo de la regla. Codex CLI no tiene una solicitud nativa, así que recibe un mensaje con el motivo, y la llamada continúa.

Optimize

Optimize cambia el camino de una acción en lugar de pararla. Cada cambio lo hace una palanca, y lo que hace una palanca depende del modo de la política.

Palanca	Estado
loop_stop	Actúa en Enforce; mide en Monitor
steer	Actúa en Enforce; mide en Monitor
narrow_output	Actúa en Enforce en Claude Code; mide en Monitor
effort_clamp	Mide por defecto, tanto en Monitor como en Enforce; actúa en Enforce solo donde lo actives
context_edit	Mide por defecto, tanto en Monitor como en Enforce; actúa en Enforce solo donde lo actives
prefix_guard	Mide por defecto, tanto en Monitor como en Enforce; actúa en Enforce solo donde lo actives
substitute	Esta palanca no está disponible, y en Monitor solo mide

En Monitor, una palanca solo mide lo que habría hecho, y el registro lo indica. En Enforce, se aplica.

Una acción, cuatro veredictos

Escenario ilustrativo. La misma acción, `git push`, puede recibir cada veredicto según su contexto:

Contexto	Veredicto
Un agente hace push de una rama de funcionalidad de un repositorio que es de su equipo	Allow
Un agente hace push a una rama de release compartida	Ask: el push se ejecuta y se marca para revisión
Un agente fuerza un push a la rama principal de un repositorio de producción	Block
Un agente reintenta en bucle el mismo push que falla	Optimize: en <code>Enforce</code> , <code>loop_stop</code> termina el bucle

 Nota

Ask evita la repetición, no la primera vez. En Codex CLI aparece como un mensaje y la llamada continúa.

Modos

Cada política de intención se ejecuta en uno de tres modos en cada zona que la contiene. El modo decide si una coincidencia solo se registra o si su veredicto se aplica antes de que se ejecute la acción.

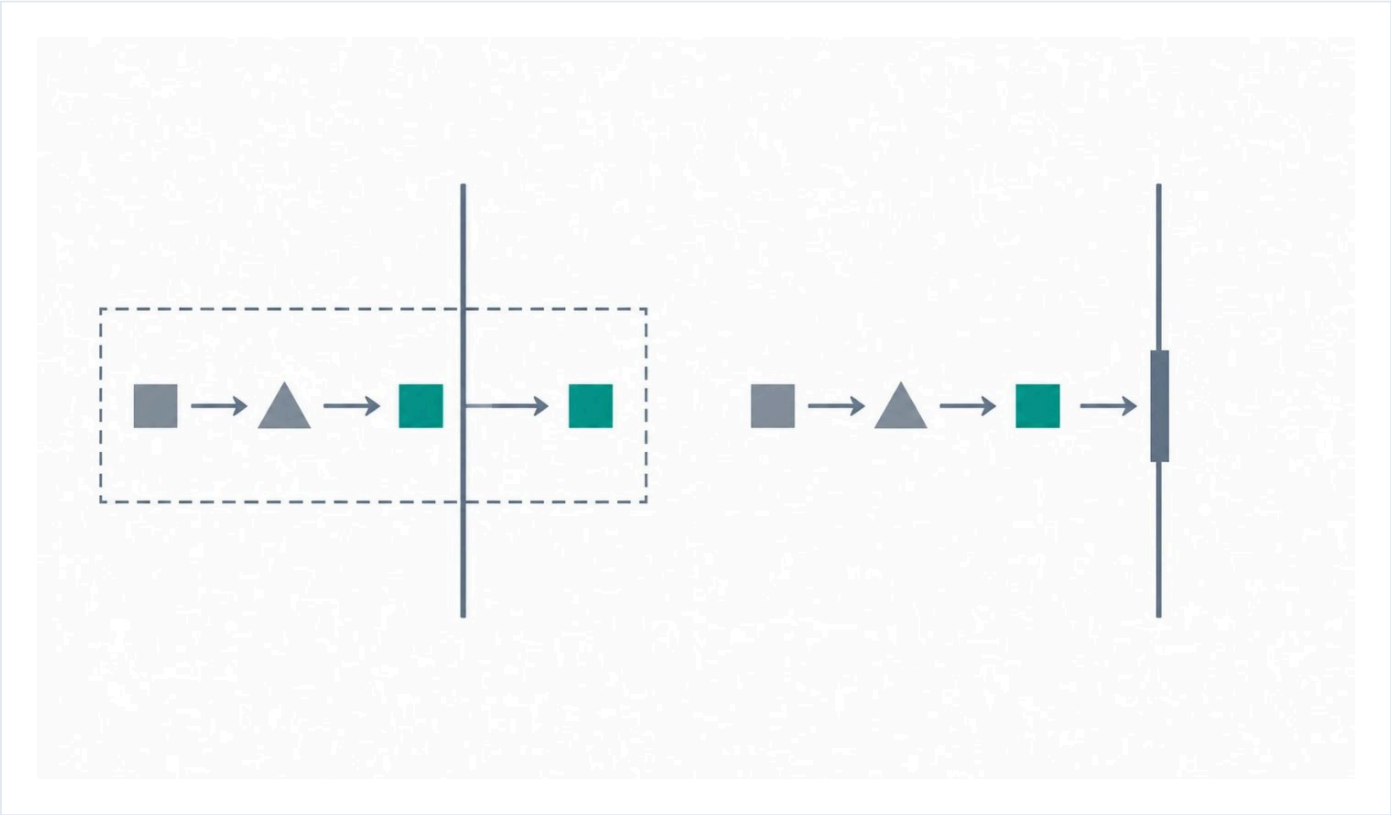


Figura 1. Las mismas acciones en Monitor y en Enforce: en Monitor cruzan el límite y quedan registradas; en Enforce, el veredicto se aplica primero.

Modo	Qué hace
Disabled	La política se conserva, con su historial, pero no se evalúa nada
Monitor	Una coincidencia se registra y se marca, y la acción continúa
Enforce	El cliente aplica el veredicto de la política antes de que se ejecute la acción, y registra si lo hizo

El modo pertenece a la política en una zona, nunca a un átomo suelto. Hay una excepción integrada en el propio átomo: un átomo cuya clase es `monitor-only` se puede observar pero no aplicar, así que se queda en

Monitor incluso cuando su política se ejecuta en Enforce.

Ganarse Enforce con evidencia

Empieza una política nueva en Monitor. Registra cada acción que habría decidido, sin cambiar lo que hace el agente. Lee sus anomalías: cuando muestren que la política dice lo que querías decir, pásala a Enforce. Cuando muestren lo contrario, editala o desactívala.

OpenLatch nunca amplía por su cuenta la autoridad de un agente. Cada paso de Monitor a Enforce, y cada política nueva, es una decisión que toma alguien de tu organización.

Una política, varias zonas

La misma política puede estar en varias zonas, cada una con su propio modo, por ejemplo Monitor en un equipo y Enforce en otro, sin ninguna copia que mantener sincronizada.

Regla	Qué significa
Gana el más estricto	Cuando el camino de un agente contiene la misma política más de una vez, se ejecuta una sola vez, en el modo más estricto: Enforce antes que Monitor, antes que Disabled
Una zona solo endurece	Una política solo se activa en una zona si no relaja las zonas superiores. Mantenerla en Disabled siempre está permitido
Bajar nunca se bloquea	Pasar una política de Enforce a Monitor, o a Disabled, siempre está permitido
Una sola versión en todas partes	Cada zona ejecuta la versión vigente de la política, cada una en su propio modo

Anomalías

Una **anomalía** es una acción de un agente que OpenLatch determina que infringe una política de intención de una de las dimensiones de la Autonomy Zone del agente. Es lo único que señala OpenLatch.

Cada anomalía indica la política que infringió, la dimensión de esa política y la zona de la que proviene la política. Se explica por construcción: la produjo una regla compilada a partir de una política que tu organización ratificó, y nada más.

Qué hace una anomalía en cada modo

Modo de la política	Qué ocurre
Disabled	No se evalúa nada, así que no se genera ninguna anomalía
Monitor	La anomalía se registra y se marca, y la acción continúa
Enforce	La acción recibe su veredicto antes de ejecutarse, y la anomalía lo registra

Dónde verlas

La página **Anomalies** de la consola enumera todas las anomalías de tu organización. Cada agente tiene además una pestaña **Anomalies** solo con las suyas.

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Anomalies

Ctrl+K

?

99+

DC

Anomalies

Review actions that crossed an intent policy, then archive the evidence when triage is complete.

Waiting on your answer

All

WAITING ON YOUR ANSWER

0

nothing waiting

ANOMALIES - 7 DAYS

26

across 10 agents

BLOCKED

50

ALLOWED AND FLAGGED

50

Nothing is waiting on your answer

Every flagged action has an answer against it, so the policy knows what to do next time.

Export to CSV

La página Anomalies, que enumera las acciones que infringieron una política de intención, con la política, su dimensión y el veredicto.

Lo que no son las anomalías

Las anomalías provienen solo de las políticas de intención. OpenLatch no señala una acción porque difiera de lo que un agente suele hacer: si ninguna política ratificada cubre una acción, no genera ninguna anomalía. Para señalar algo nuevo, escribe la política de intención que lo diga.

Grafo de ejecución

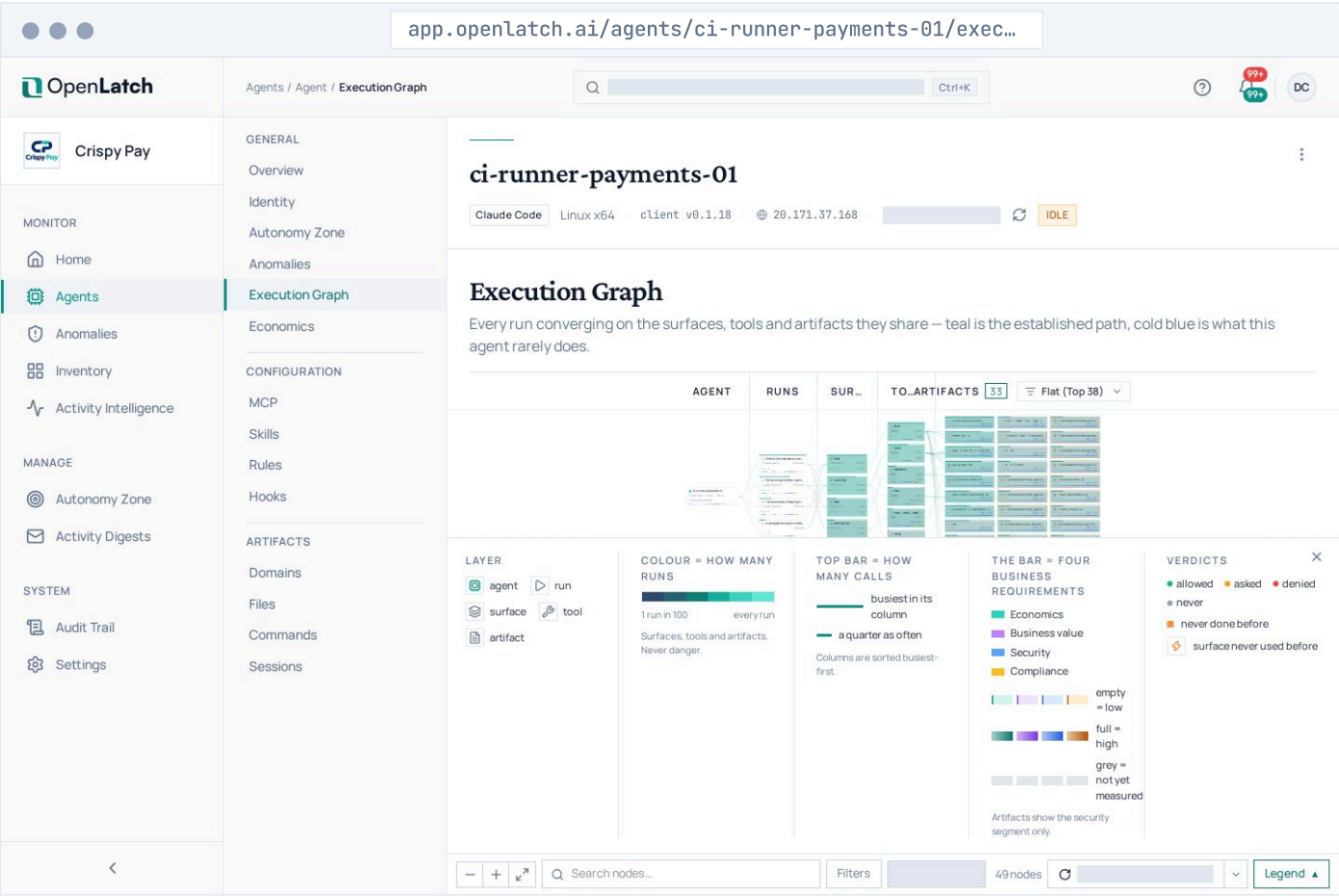
El grafo de ejecución es el registro de lo que un agente hizo realmente. Se construye a partir de los eventos que el cliente envía después de cada decisión, y es lo que OpenLatch contrasta con la Autonomy Zone del agente.

Qué muestra

Cada agente tiene una pestaña **Execution Graph**. Se lee de izquierda a derecha, desde el agente hasta lo que tocó:

Carril	Qué contiene
Agente	El agente al que se refiere el grafo
Ejecuciones	Sus sesiones, una tarjeta por ejecución
Superficies	A qué recurrió, deducido de los nombres de las herramientas
Herramientas	Lo que invocó
Artefactos	Lo que tocó o ejecutó: archivos, dominios de red y comandos

Los carriles convergen en lo que las ejecuciones comparten, así que ves los caminos que un agente sigue más y los que sigue rara vez.



La pestaña Execution Graph de un agente: ejecuciones, superficies, herramientas y artefactos, de izquierda a derecha.

Dos planos lo alimentan

Plano	Qué captura el cliente
Plano de comandos	Cada llamada a una herramienta y cada comando de shell, interceptados en el hook nativo del agente antes de ejecutarse
Plano de solicitudes	Cada solicitud saliente a un modelo, interceptada por un listener en la misma máquina, con su modelo, sus tokens y su coste

Las demás pestañas de un agente desglosan el mismo registro por tipo: comandos, archivos, dominios, servidores MCP, skills, hooks, sesiones y economía.

De una acción a una política

Cuando el grafo muestre un comando que quieras controlar, ábrelo y crea una nueva política de intención a partir de esa acción. El brief parte de lo que el agente ejecutó realmente, y la política pasa por la misma revisión y ratificación que cualquier otra. [Políticas de intención](#) explica ese camino.

Una acción que ninguna política decidió se registra como permitida por defecto, no como autorizada por una decisión. El grafo distingue ambos casos.

Niveles de control

Cada agente del registro de agentes tiene un nivel de control. Te dice lo que OpenLatch puede hacer realmente con ese agente, para que nunca tengas que suponer que un agente está controlado porque aparece en la lista.

Nivel de control	Qué significa
Enforced	El cliente está en el camino de ejecución del agente. Un veredicto llega al agente y se respeta
Monitored	La integración está instalada y los eventos fluyen, pero todavía no se puede entregar un veredicto, así que no se impide nada
Known	El agente está identificado y clasificado, y se indican los límites del control

La cobertura es un resultado del despliegue. El nivel de control es lo que el despliegue consiguió para ese agente, no una promesa sobre todos los agentes de su tipo.

Dos planos

Un agente se cubre en dos planos, y el nivel de control siempre se indica por plano.

Plano	Qué intercepta	Qué puede llevar hoy
Plano de comandos	Una llamada a una herramienta o un comando de shell, interceptado en el hook nativo del agente	Allow, Ask, Block u Optimize, decidido antes de que se ejecute la acción. Optimize empieza en Monitor
Plano de solicitudes	Una solicitud saliente a un modelo, interceptada por un listener en la misma máquina	Allow y registro por defecto, así que es Observed: capturado y medido. Ask no tiene sentido aquí, y Block no está disponible en este plano

Agentes conectados hoy

Agente	Distribución	Plano de comandos	Plano de solicitudes
Claude Code	Cualquiera	Enforced en el plano de comandos	Observed en el plano de solicitudes
Codex CLI	Gestionada, mediante MDM, un ajuste del sistema o una configuración en la nube	Enforced en el plano de comandos, y el desarrollador no puede desactivarlo	Observed en el plano de solicitudes
Codex CLI	Autoservicio	Monitored en el plano de comandos hasta que el desarrollador confía en el hook de OpenLatch dentro de Codex CLI, y después Enforced en ese plano	Observed en el plano de solicitudes
Cursor	Autoservicio, en el IDE y la CLI, en macOS y Windows	Enforced en el plano de comandos	Observed en el plano de solicitudes
Cline	Autoservicio, en VS Code, en macOS	Monitored en el plano de comandos	Observed en el plano de solicitudes

`openlatch status` y `openlatch doctor` indican cada agente de la máquina con su nivel de control.

Cómo recibe cada agente un veredicto

No todos los agentes pueden expresar de forma nativa todos los veredictos. Un veredicto que un agente no puede expresar se entrega como el más cercano que sí puede, se registra como tal y nunca es más estricto de lo que pretendía la regla.

Agente	Ask	Block
Claude Code	Su propia solicitud nativa	Su rechazo nativo
Codex CLI	Un mensaje con el motivo de la regla; la llamada continúa	Su rechazo nativo
Cursor	Nativo en comandos de shell y llamadas MCP en algunos modos de ejecución del IDE; en el resto, la acción se permite y el registro indica que no se pudo entregar Ask	Su rechazo nativo

Cómo funciona OpenLatch

Esta página sigue una acción desde el momento en que el agente la intenta hasta el momento en que su registro llega a la plataforma. Las políticas se escriben una vez, en la plataforma. Cada decisión se toma en la máquina, por el cliente, sin ninguna llamada de red.

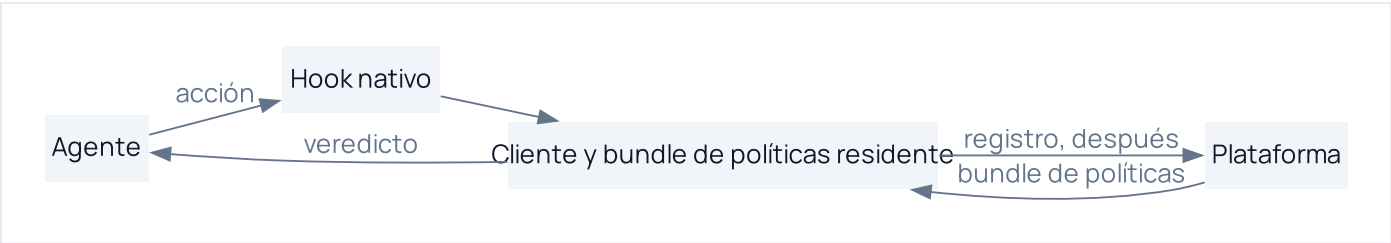


Figura 1. La plataforma compila y distribuye las políticas. El cliente decide localmente cada acción cubierta y después envía su registro a la plataforma.

1. Las políticas llegan al cliente

Ratificas políticas de intención en la consola. La plataforma las compila y construye un bundle de políticas por agente, a partir de la zona efectiva de ese agente. El cliente pide su bundle a la plataforma de forma periódica:

La plataforma responde	El cliente
Nada ha cambiado	Conserva el bundle que tiene
Un bundle nuevo	Comprueba su integridad y luego lo sustituye entero
Nada, o un bundle que no supera la comprobación	Conserva el último bundle válido

Para obtener el bundle de inmediato, ejecuta `openlatch system aip refresh`.

2. El hook intercepta la acción

El cliente se conecta a los hooks nativos del propio agente. Cuando el agente intenta una acción cubierta, como ejecutar un comando de shell o llamar a una herramienta, el hook se la pasa al cliente antes de que se

ejecute. Las solicitudes salientes a un modelo pasan por un listener en la misma máquina.

3. El cliente decide

El cliente evalúa la acción con su bundle residente, en memoria, en la máquina. En este camino no hay ninguna llamada de red, así que el agente nunca depende de la red y el veredicto tampoco. Cuando coinciden varias políticas, gana el veredicto más restrictivo. [Veredictos](#) y [Modos](#) explican lo que hace cada veredicto en Monitor y en Enforce.

4. El registro llega después

Después de la decisión, el cliente oculta las credenciales en la máquina y envía a la plataforma el registro de la acción y su veredicto. La plataforma conserva ese registro para el registro de auditoría, el grafo de ejecución y los informes. Nunca vuelve a decidir y nunca cambia un veredicto que el cliente ya devolvió.

Sin conexión

El bundle residente es la autoridad, no una caché. Con la red desconectada, el cliente sigue aplicando el último bundle válido, de forma indefinida. Obtener políticas nuevas y enviar registros necesitan conexión; decidir, no.

Dos situaciones dejan una acción sin decidir, y entonces la acción continúa: la máquina nunca ha tenido un bundle válido, o el hook no puede comunicarse con el cliente en su propia máquina. `openlatch` doctor informa de ambas.

Privacidad y telemetría

Esta página explica lo que sale de tu máquina y de nuestros sitios, y cómo desactivar lo que no quieres que se envíe.

Lo que el cliente envía a la plataforma

Después de cada decisión, el cliente envía a la plataforma el registro de la acción y su veredicto: lo que el agente intentó hacer, como un comando, una ruta de archivo, una herramienta o un destino de red, y lo que se decidió. Ese registro es lo que la consola muestra en el grafo de ejecución, las anomalías y el registro de auditoría.

- Las credenciales se ocultan en la máquina antes de que nada salga de ella.
- El cliente también mantiene un registro de auditoría local en la máquina.
- La decisión en sí nunca depende de la plataforma: consulta [Cómo funciona OpenLatch](#).

Telemetría de uso del cliente

El cliente envía telemetría de uso e informes de fallos para ayudarnos a mejorarlo. Mientras el cliente no ha iniciado sesión, la telemetría de uso solo lleva un identificador aleatorio de la instalación. Una vez iniciada la sesión, el cliente vincula ese identificador a los identificadores de tu cuenta y de tu organización, así que su telemetría de uso deja de ser anónima. Ambos van a la propia dirección de la plataforma, así que no añaden ningún destino a tus reglas de red. Puedes desactivar cualquiera de los dos.

Para	Haz esto
Ver si la telemetría de uso está activada, y por qué	<code>openlatch system telemetry status</code>
Desactivar la telemetría de uso	<code>openlatch system telemetry disable</code> , o define <code>OPENLATCH_TELEMETRY_DISABLED=1</code> o <code>DO_NOT_TRACK=1</code>
Volver a activar la telemetría de uso	<code>openlatch system telemetry enable</code>
Desactivar los informes de fallos	Define <code>OPENLATCH_CRASH_REPORTING=0</code>

Analítica de openlatch.ai y de esta documentación

`openlatch.ai` y `docs.openlatch.ai` no tienen inicio de sesión y no cargan rastreadores de terceros. Registran analítica de producto anónima mediante [PostHog](#), alojado en la UE, a través de una ruta propia en `openlatch.ai`.

Registramos:

- las páginas que visitas y cuánto tiempo te quedas;
- eventos de conversión, como copiar un comando de instalación o seguir un enlace a la consola;
- las Core Web Vitals, para mantener los sitios rápidos;
- una muestra de repeticiones de sesión, con todo el texto y los campos de formulario ocultos;
- los errores de JavaScript no controlados, después de que un filtro en tu navegador sustituya todo lo que parezca una dirección de correo o un token de API.

Nunca recogemos lo que escribes en un formulario, nunca te identificamos desde estos sitios y no vendemos ni compartimos estos datos.

Darse de baja

Añade `?ph_opt_out=1` a la dirección de cualquier página para desactivar la analítica en este dispositivo. La elección se guarda en tu navegador y se mantiene al recargar. También puedes bloquear la ruta `/ingest/` con cualquier bloqueador de contenido.

Al pasar a la consola

Cuando sigues un enlace de estos sitios a `app.openlatch.ai`, el identificador anónimo viaja en el fragmento de la dirección, después del #, para que la consola pueda unir tus visitas anteriores a tu cuenta cuando te registres. El fragmento nunca llega a nuestros servidores.

Guías

Cada guía cubre una tarea en la consola de `app.openlatch.ai`, con la página en la que la haces. Si empiezas, comienza por [Escribe tu primera política](#); para entender por qué la consola se comporta así, lee los [Conceptos](#).

Políticas y zonas

Guía	Páginas de la consola
Redactar una Agent Intent Policy	Autonomy Zone > Policies , el asistente de creación de una política de intención
Pasar una política a Enforce	Autonomy Zone > Policies
Gestionar las Autonomy Zones	Autonomy Zone y sus pestañas Agents y Audit
Revisar las anomalías	Anomalies

Agentes

Guía	Páginas de la consola
Encontrar y leer un agente	Agents , y las pestañas Overview , Identity y Autonomy Zone de un agente
Investigar lo que hizo un agente	Las pestañas Sessions , Execution graph , Commands , Files , Domains y Anomalies de un agente
Revisar la configuración de un agente	Las pestañas Hooks , MCP servers , Rules y Skills de un agente
Seguir el gasto de un agente	La pestaña Economics de un agente
Buscar en el inventario	Inventory y sus pestañas
Seguir los eventos en directo	Events
Leer la página de inicio	Home

Informes

Guía	Páginas de la consola
Programar resúmenes de actividad	Digests
Configurar los canales de notificación	Settings > Notification Channels
Revisar el registro de auditoría	Audit Trail , Settings > Audit Trail

Ajustes de la organización

Guía	Páginas de la consola
Gestionar usuarios y equipos	Settings > Users, Settings > Teams
Configurar el inicio de sesión único	Settings > SAML SSO, Settings > Auth Providers
Conectar un directorio para buscar propietarios	Settings > Identity Providers
Editar el perfil de la organización	Settings > Organization Profile
Ajustar el catálogo	Settings > Catalogue
Consultar licencias y uso	Settings > Licensing

Tu cuenta

Guía	Páginas de la consola
Gestionar las claves de API	Account > API Keys
Gestionar tu cuenta	Account > Profile, Account > Security

Desplegar OpenLatch en un parque tiene su propia sección: [Desplegar](#).

Redactar una Agent Intent Policy

Redacta una Agent Intent Policy (AIP) cuando quieras que una Autonomy Zone permita, cuestione o detenga un tipo de acción de agente. Necesitas el rol editor o superior. Si el asistente indica que la redacción no está disponible, un administrador de tu sitio de OpenLatch aún no ha configurado la redacción asistida por IA.

Una AIP es un conjunto de átomos de intención. A partir de ellos, OpenLatch genera una frase y casos de prueba concretos. Ratificas la frase, con Sí o No, con los casos de prueba mostrados al lado; nunca editas las reglas compiladas. [Políticas de intención](#) explica el modelo.

Crear la política

- 1 Abre **Autonomy Zone**, selecciona la unidad cuya zona tendrá la política, abre **Policies** y empieza a crear una nueva política de intención.
- 2 **Brief.** Di lo que quieres en una o dos frases, con tus propias palabras, o parte de un ejemplo.
- 3 **Policy.** Lee la frase que generó OpenLatch, con los casos de prueba al lado. Cada caso es una situación concreta y el veredicto que devolvería la política. Responde Sí si la frase dice lo que querías decir. Responde No y di lo que querías decir en su lugar: OpenLatch regenera la frase y sus casos.
- 4 **How it runs.** Elige el modo en que se ratifica la política: Disabled, Monitor o Enforce. Empieza en Monitor salvo que ya confíes en la política. Donde una regla pueda preguntar a una persona, elige qué ocurre mientras llega la respuesta: por defecto, la acción se ejecuta y se marca para revisión; para una acción irreversible puedes activar, por átomo, que se detenga hasta que alguien responda.
- 5 **Ratify.** Ejecuta la reproducción, que aplica la política a la actividad reciente de tus agentes y muestra lo que habría decidido, y ratifica. No se aplica nada hasta que lo haces.

app.openlatch.ai/zone/policies/new

OpenLatch

Autonomy Zone / New intent policy

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

New intent policy

Turn what you already have into an intent policy. Nothing runs until you ratify it.

1

Brief

2

Policy

3

How it runs

4

Ratify

Define the policy intent

Provide a brief or reference material. OpenLatch will use it to draft an Agent Intent Policy for your review.

Paste a brief

Type or paste what you mean, in your own words.

Upload a policy

A document, an SOP, a procedure. We read it and pull the intents out.

Record a conversation

Describe the rule out loud, or attach the meeting where it was agreed.

Your brief

Examples

e.g. Payments engineering change-control SOP:

- No production deployment or database migration without an approved change ticket.

- During declared incidents, the on-call engineer may act without a ticket, recording actions on the incident bridge.

- Destructive commands (DROP TABLE, rm -rf, kubectl delete namespace) are strictly forbidden on production clusters.

- All access to customer secrets or payment credentials must require an active session.

One rule per line. Anything we infer is marked as inferred, on the line itself.

Write what the agent does, not what it intends

Name the exact action: command, path, domain or tool (e.g. "Block Bash when command.program equals rm").

Say the outcome: block it, or ask a human before it runs.

Phrase exceptions as a fact the agent shows: "unless agent.environment equals development".

This policy will be created in Crispy Pay.

Turn it into a policy

El asistente de creación de una política de intención: Brief, Policy, How it runs, Ratify.

Ratificar reconstruye el paquete de políticas. Cada cliente lo obtiene en su siguiente consulta; para obtenerlo al momento en una máquina, ejecuta `openlatch system aip refresh`.

Gestionar las políticas en la lista

La pestaña **Policies** enumera las políticas de intención redactadas en la unidad seleccionada, con sus dimensiones, su capa, su modo y las anomalías que generó cada una en los últimos siete días. Activa **Show inherited policies** para ver las políticas que una unidad hereda de las capas superiores.

app.openlatch.ai/zone/policies

OpenLatch

Autonomy Zone / Policies

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Autonomy Zone

The operating envelope an agent works inside – what it may do, under which conditions, and which consequences the business accepts. The deployed client evaluates each covered action locally before it runs; OpenLatch records the outcome.

Q Find a unit... Add unit

Unit (50) Autonomy Zone (18)

Crispy Pay +9 151

Corporate 49

Finance

IT

Legal

People +2 1

Security +3 8

AppSec

GRC

SecOps

Detection & Response

Data & Analytics +1 6

BI

MLPlatform

Merchant Success 4

Onboarding

Support Engineering

No directory connected

Crispy Pay

Defines its own Autonomy Zone

Autonomy Zone Policies (9) Agents (0) Audit

Add policies Add policy preset

The intent policies attached to Crispy Pay.

Q Search intent policies

Dimension Mode

INTENT POLICY	DIMENSION	LAYER	MODE	ANOMALIES 7D
☐ The agent only reaches appr... AIP-001 OWASP Agentic Security - ve...	Security	Attached here	MONITOR	11
☐ No destructive data-store o... AIP-002 OWASP Agentic Security - ve...	Safety	Attached here	ENFORCE	9
☐ Secret stores are never read AIP-003 OWASP Agentic Security - ve...	Security	Attached here	ENFORCE	13
☐ Opaque downloaded code is ... AIP-004 OWASP Agentic Security - ve...	Security	Attached here	ENFORCE	11
☐ Granting a file the execute b... AIP-005	Security	Attached here	MONITOR	0

La pestaña Policies: cada política de intención con su dimensión, su capa, su modo y sus anomalías recientes.

Para	Haz esto
Leer una política	Ábrela. El panel muestra sus intenciones, los casos de prueba generados, su actividad, su procedencia y el historial de sus versiones ratificadas
Cambiar su modo	Usa Mode en las acciones de la fila, o selecciona varias políticas y elige Enforce , Monitor o Disable
Cambiar una política heredada	Abre la capa que la redactó. Las acciones en bloque solo se aplican a las políticas redactadas en la unidad seleccionada
Partir de plantillas	Elige Add policy preset , revisa cada política de la plantilla, elige el modo de cada una y aplícalas
Eliminar una política	Elige Delete . La política se retira para esta capa y todas las inferiores, y su historial se queda en la auditoría de la zona

Una política cuyas intenciones son todas solo de Monitor puede ejecutarse en Monitor pero no en Enforce: Monitor registra lo que ocurre, y Enforce necesita al menos una intención aplicable.

Dejar que la cobertura ajuste una política

OpenLatch aprende qué herramientas, programas y destinos intervienen en las acciones de tus agentes, y puede proponer cambios que mantengan una política cubriendo aquello para lo que se escribió. El

Coverage mode de la política decide qué pasa con esas propuestas:

Coverage mode	Qué ocurre
Off	La cobertura no se analiza para esta política
Strict	El aprendizaje se cuenta y se muestra; no se propone ni se aplica nada
Explicit	Las propuestas esperan tu revisión
Auto	Los cambios restrictivos y las ampliaciones del catálogo se aplican automáticamente, y cada uno queda escrito en el historial de la política

Revisa las propuestas desde la sección **Coverage** de la política, apruébalas o descártalas y aplícalas, lo que ratifica una nueva versión. [Ajustar el catálogo](#) cubre las entradas que aprende la cobertura.

Después, [pasa la política a Enforce](#) cuando su historial muestre que es acertada.

Pasar una política a Enforce

Ejecuta una política en Monitor hasta que su historial muestre que decide lo que esperas, y luego pásala a Enforce. Necesitas el rol editor o superior, en la unidad que redactó la política.

En Monitor, una política registra sus anomalías y nunca interviene. En Enforce, el cliente aplica su veredicto antes de que la acción se ejecute. [Modos](#) explica la diferencia.

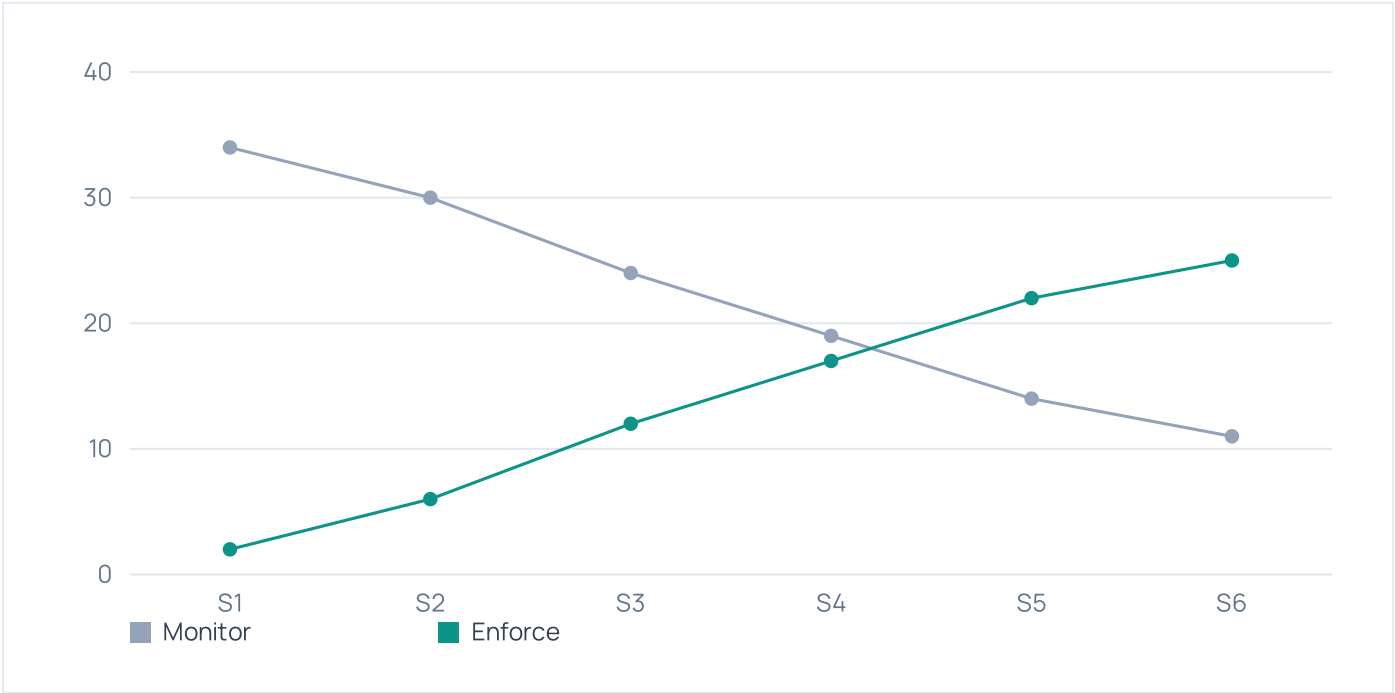


Figura 1. Las políticas de Crispy Pay en sus seis primeras semanas: la mayoría pasan de Monitor a Enforce cuando sus veredictos son fiables. Datos de demostración ilustrativos.

Decidir cuándo pasar a Enforce

Pasa una política a Enforce cuando su historial en un periodo representativo, una semana de trabajo normal para la mayoría de los equipos, no contenga sorpresas:

Comprobación	Dónde
Cada anomalía que generó es una acción sobre la que quieres que actúe	La Activity de la política y, después, Anomalies filtrado por la política
No generó ninguna anomalía en un trabajo que consideras normal	La misma lista, leída con los propietarios de los agentes
Sus intenciones alcanzan el umbral de aplicación	El panel de la política. Una intención marcada como solo de Monitor genera anomalías y nunca interviene, en ningún modo

Cuando el historial muestre un falso positivo, cambia la política en lugar de pasarla a Enforce: ábrela, responde No a su frase diciendo lo que querías decir y ratifica la nueva versión en Monitor.

Cambiar el modo

- 1 Abre **Autonomy Zone**, selecciona la unidad que redactó la política y abre **Policies**.
- 2 En la fila de la política, abre sus acciones y pon **Mode** en **Enforce**. Para pasar varias a la vez, selecciónalas y elige **Enforce**.
- 3 Cada cliente aplica el cambio en su siguiente consulta. Para aplicarlo al momento en una máquina, ejecuta `openlatch system aip refresh`.

Desde entonces, el cliente decide las intenciones aplicables de la política antes de que la acción se ejecute: Allow, Ask, Block u Optimize. Una palanca de Optimize solo actúa en Enforce; en Monitor mide lo que habría hecho. El modo de cada política queda en el registro de auditoría con quién lo cambió.

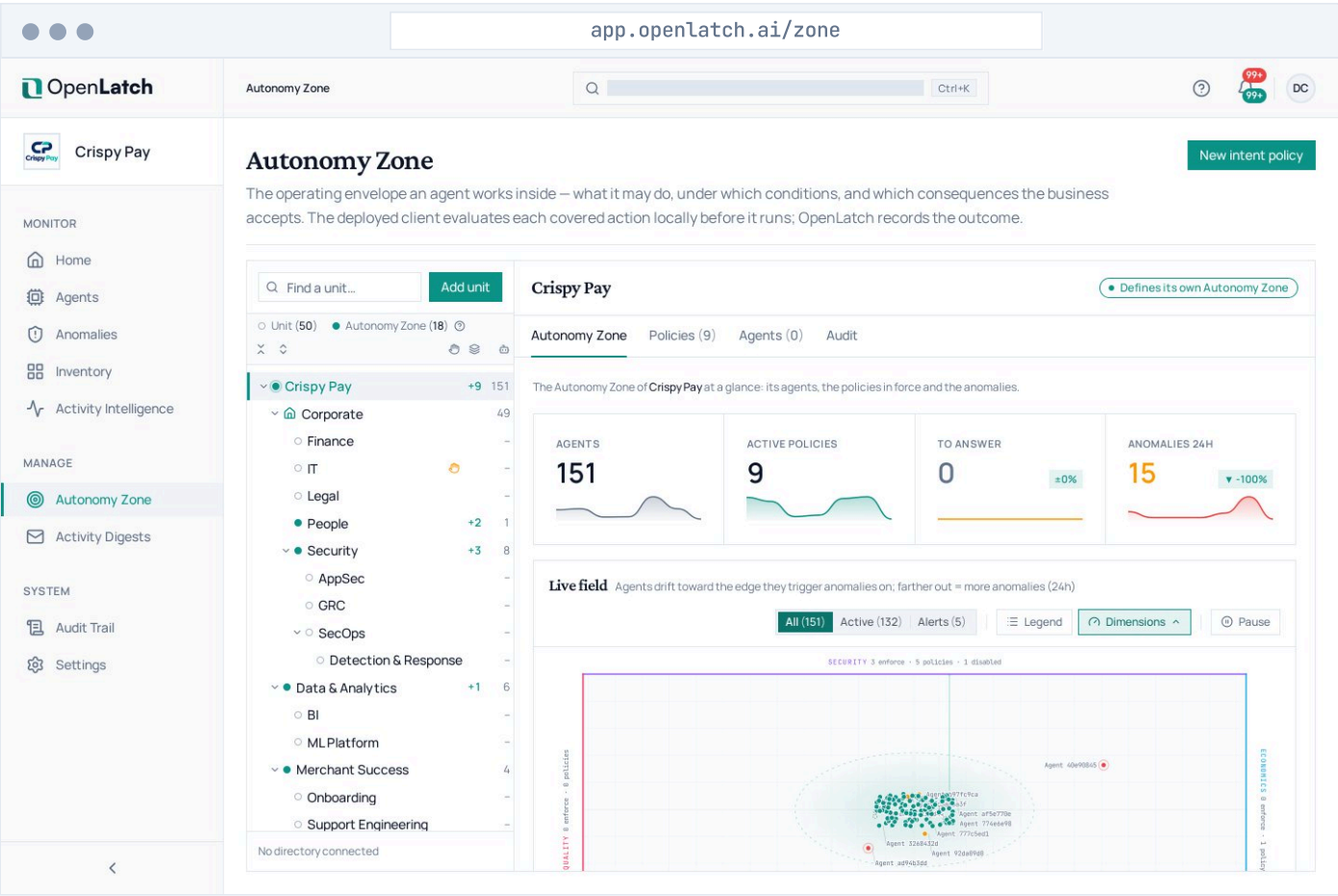
Volver atrás

Para hacer volver una política, pon su modo en **Monitor**. Sigue registrando sus anomalías y deja de intervenir en la siguiente consulta. Ponla en **Disabled** para dejar de evaluarla por completo; su historial sigue disponible.

Gestionar las Autonomy Zones

Usa la página **Autonomy Zone** para decidir qué partes de tu organización tienen sus propias reglas, qué agentes están dónde y qué decidió cada zona. Necesitas el rol editor o superior para cambiar una zona; cualquiera puede leerla.

Las Autonomy Zones se sitúan en el árbol de unidades de tu organización. Cualquier unidad puede tener una zona, y la mayoría no la tiene: una unidad sin zona hereda la zona más cercana por encima de ella, y cada agente se resuelve en una única zona efectiva. [Autonomy Zone](#) explica el modelo.

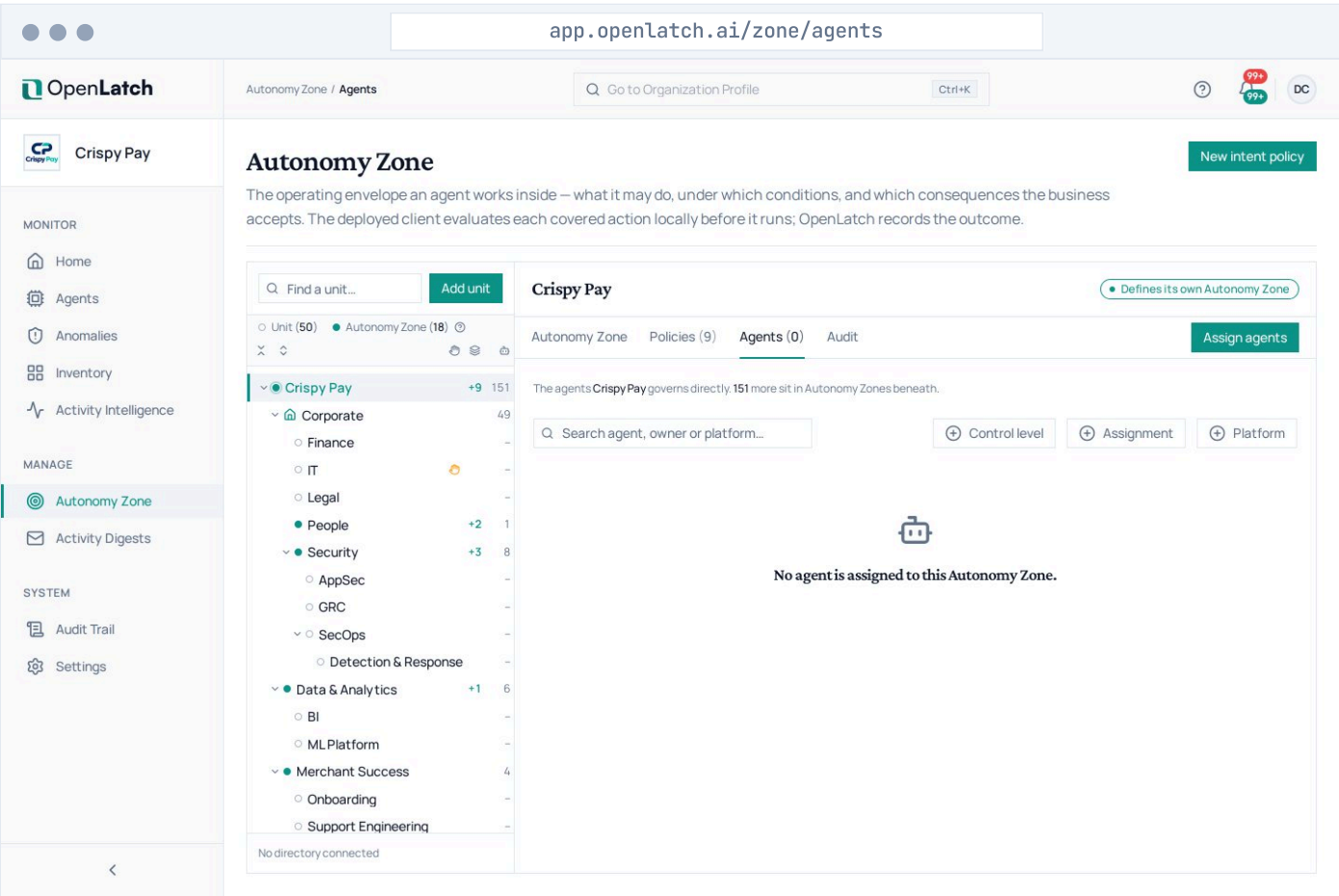


La página Autonomy Zone: el árbol de unidades de la organización y la zona de la unidad seleccionada de un vistazo.

Leer una zona

Selecciona una unidad en el árbol. La página muestra su zona de un vistazo: sus agentes, las políticas de intención vigentes y las anomalías que generaron. Las pestañas van más allá:

Pestaña	Qué muestra
Policies	Las políticas de intención redactadas en la unidad y, si lo pides, las que hereda. Consulta Redactar una Agent Intent Policy .
Agents	Los agentes que la unidad gobierna directamente
Audit	Cada acción que intentó un agente de la unidad, con la regla que coincidió y la decisión



La pestaña Agents: los agentes que una unidad gobierna directamente.

The screenshot shows the OpenLatch application interface. The top navigation bar includes the OpenLatch logo and a search bar. The left sidebar contains a 'MONITOR' section with links to Home, Agents, Anomalies, Inventory, and Activity Intelligence, and a 'MANAGE' section with links to Autonomy Zone, Activity Digests, Audit Trail, and Settings. The main content area is titled 'Autonomy Zone' and includes a description: 'The operating envelope an agent works inside – what it may do, under which conditions, and which consequences the business accepts. The deployed client evaluates each covered action locally before it runs; OpenLatch records the outcome.' Below this, there's a section for 'Crispy Pay' with a list of units and their counts. The 'Audit' tab is selected, showing a table of runtime decisions. The table has columns for WHEN, AGENT, DIMENSION, MATCHED, ACTION VALUE, MODE, and DECISION. The table is currently empty, showing a message: 'No runtime decisions in this window. Decisions appear here after a client reports evaluated activity for this Autonomy Zone.'

La pestaña Audit: cada acción que intentó un agente de la unidad, con la regla que coincidió y la decisión.

Dar a una unidad su propia zona

- 1 Selecciona la unidad y abre sus ajustes.
- 2 Activa su propia Autonomy Zone. La zona empieza vacía y se comporta exactamente como la zona superior hasta que redactes una política en ella: cada unidad hereda todas las capas superiores en cualquier caso.
- 3 Redacta las políticas que solo se aplican aquí. Consulta [Redactar una Agent Intent Policy](#).

Para quitar a una unidad su propia zona, usa **Delete Autonomy Zone** en los ajustes de la unidad. Las políticas redactadas en ella se retiran, y sus agentes pasan a la zona superior.

Añadir o cambiar una unidad

Las unidades suelen venir de tu directorio, mediante [Conectar un directorio para buscar propietarios](#). El nombre y la posición de una unidad del directorio pertenecen al directorio y se sobrescriben en la siguiente

sincronización; su Autonomy Zone es tuya.

Añade una unidad a mano para una parte del negocio que el directorio no modela, como una línea de producción o una empresa conjunta. Una unidad añadida a mano nunca se sobrescribe en una sincronización, y es el único tipo de unidad que puedes eliminar. Al eliminarla, sus hijas y sus agentes pasan a su unidad madre. El árbol admite hasta diez niveles de profundidad.

Colocar los agentes

Por defecto, un agente está en la Autonomy Zone de su propietario, según la unidad del propietario en el directorio. Para colocar agentes en otro sitio, abre la pestaña **Agents** de la zona y elige **Assign agents**: el panel mueve agentes a esta zona o los devuelve a la zona de su propietario. Un agente asignado a mano se queda donde lo pusiste hasta que lo devuelvas; **Remove manual assignment** deja que el directorio vuelva a decidir su zona.

La pestaña **Autonomy Zone** del propio agente muestra la zona efectiva en la que se resuelve, capa por capa. Consulta [Encontrar y leer un agente](#).

Elegir dónde llega un Ask a una persona

ASK delivery define dónde llega a una persona un Ask de las políticas de intención de la zona:

Ajuste	Dónde aparece el Ask
Platform	En la página Anomalies
Agent	En la propia ventana de diálogo del agente, donde el agente puede preguntar; en otros casos, como Platform
Both	En la ventana de diálogo del agente y en la página Anomalies

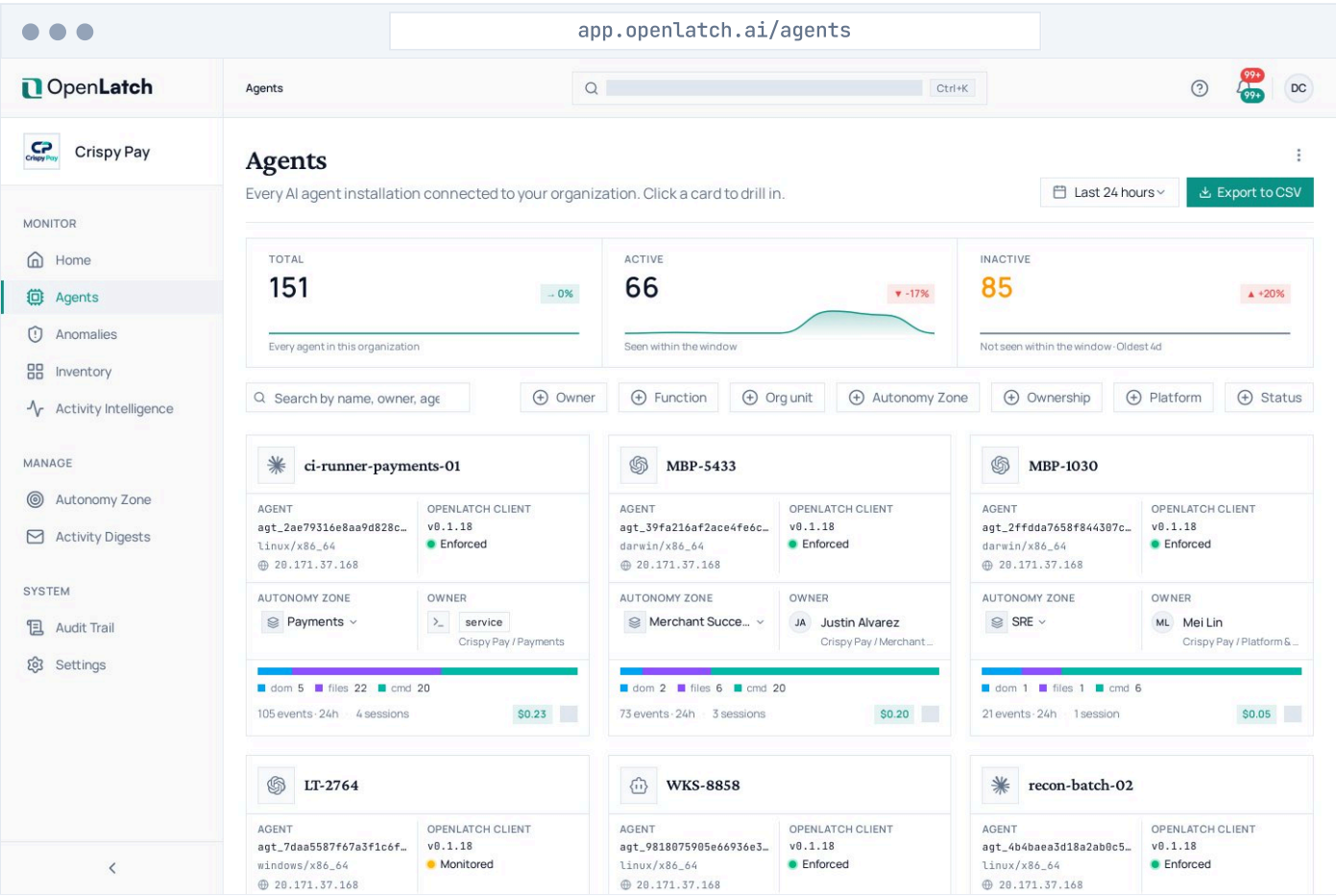
Las zonas hijas heredan el ajuste salvo que definan el suyo. Junto al ajuste se enumeran los agentes que no pueden preguntar en su propia ventana de diálogo. Por defecto, un Ask deja que la acción se ejecute y la marca para revisión. Consulta [Veredictos](#).

Encontrar y leer un agente

Usa la página **Agents** para encontrar cualquier agente que haya enviado datos a tu organización, y sus pestañas para ver para quién actúa y con qué reglas funciona. Todos los roles pueden leerla.

Encontrar un agente

La página **Agents** enumera cada agente de la organización. Las cifras de arriba cuentan todos los agentes, los agentes vistos en la ventana seleccionada y los agentes no vistos en ella. Busca por nombre, propietario, plataforma o host, y filtra la lista para acotarla.



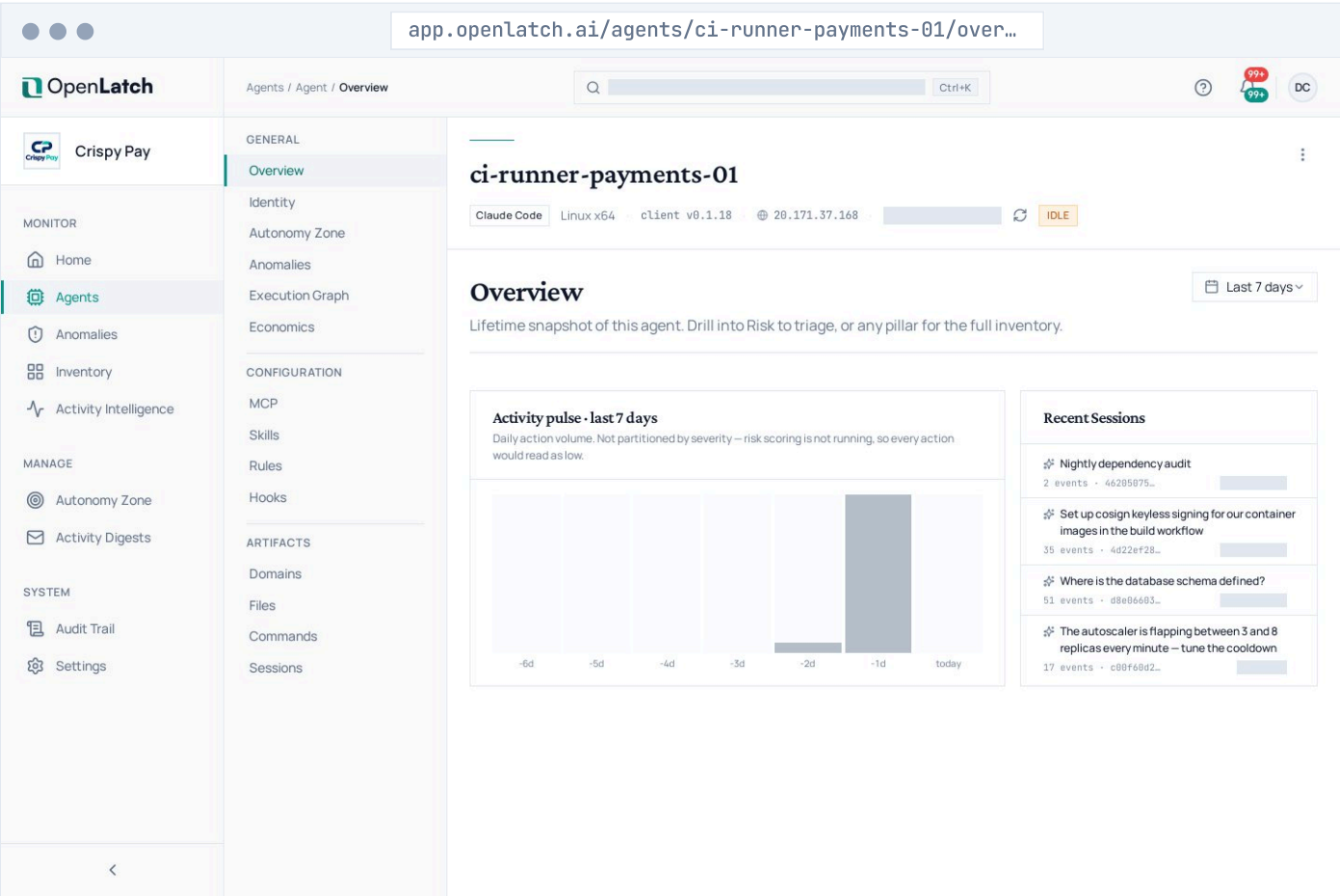
La página Agents: cada agente de la organización, con su propietario, su nivel de control y su última actividad.

Cada agente tiene un nivel de control: Enforced, Monitored o Known. Te dice qué puede hacer OpenLatch con ese agente, para que nunca supongas que un agente está controlado porque aparece en la lista.

Para conectar un agente nuevo, elige **Install an agent** y ejecuta en la máquina el comando que muestra. [Instalar el cliente](#) explica los comandos.

Leer el resumen

Abre un agente. Su pestaña **Overview** resume lo que hizo en la ventana seleccionada: sus acciones, las que fueron sensibles, los puntos de acceso a los que llegó y las anomalías que generó.



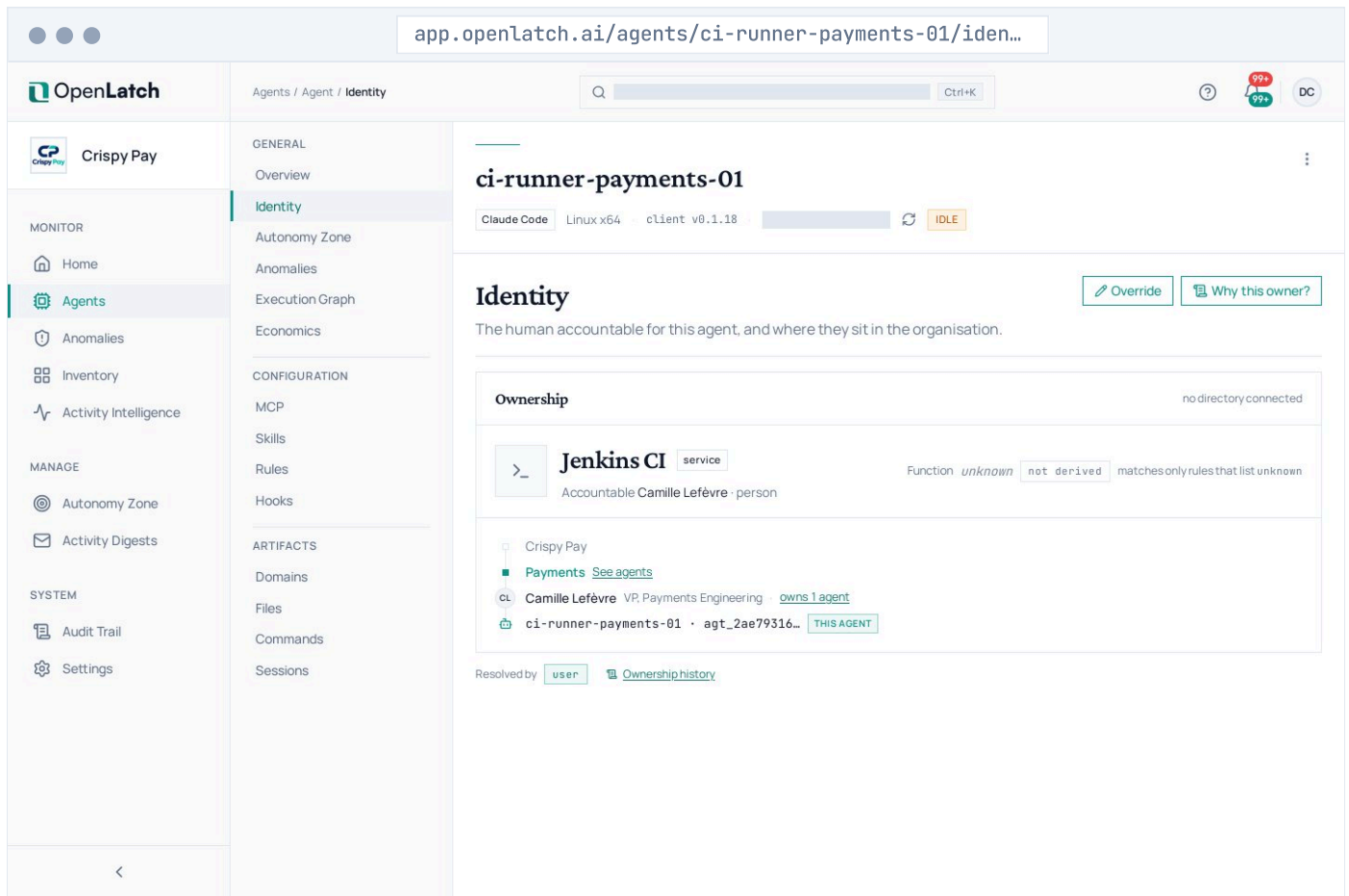
La pestaña Overview de un agente: su actividad en la ventana y lo que merece una mirada.

Las demás pestañas se tratan en [Investigar lo que hizo un agente](#), [Revisar la configuración de un agente](#) y [Seguir el gasto de un agente](#).

Comprobar quién es su propietario

La pestaña **Identity** nombra a la persona responsable del agente y su lugar en la organización. Con un directorio conectado, el propietario se determina mediante una búsqueda de solo lectura; consulta

[Conectar un directorio para buscar propietarios.](#)



La pestaña Identity de un agente: su propietario y su lugar en la organización.

- 1 Elige **Why this owner?** para ver cómo se determinó el propietario, desde las señales que envió la máquina hasta la ficha del directorio con la que coincidieron.
- 2 Si el propietario no es correcto, elige **Override ownership** y elige a la persona correcta. Si no está en el directorio, añádela a mano con su dirección de correo y, opcionalmente, su nombre y su cargo. Para un agente que actúa en nombre de un servicio y no de una persona, nombra el servicio y a la persona responsable de él.

Sustituir el propietario requiere el rol editor o superior, y queda en el registro de auditoría.

Ver su Autonomy Zone

La pestaña **Autonomy Zone** muestra la zona efectiva en la que se resuelve el agente: las capas que hereda, desde la organización hasta su propia unidad, y las políticas de intención vigentes sobre él. Un agente con nivel de control Known aparece con su zona, pero su zona no se le aplica.

The screenshot shows the OpenLATCH interface for an agent named 'ci-runner-payments-01'. The left sidebar contains navigation menus for 'MONITOR', 'MANAGE', and 'SYSTEM'. The 'MANAGE' menu is active, showing 'Autonomy Zone' and 'Activity Digests'. The main content area displays the agent's details, including its identity, configuration, and artifacts. A section titled 'Autonomy Zone' shows that 12 intent policies reach this agent from 2 Autonomy Zone layers. Below this, a table lists the effective intent policies, including one from the 'Economics' layer.

AUTONOMY ZONE	POLICIES IN FORCE	ENFORCING	ANOMALIES - 14 DAYS
Payments inherits 2 layers: Crispy Pay	12 9 inherited · 3 in its Autonomy Zo...	6 of 12 the other 6 record, and do not act	0 nothing flagged

INTENT POLICY	ALSO IN	LAYER	MODE	ANOMALIES 14D
Economics 1 policy · 0 enforcing · 1 in Monitor				
AIP-007 A session stays under sixty tool runs	-	↑ Crispy Pay	MONITOR	0

La pestaña Autonomy Zone de un agente: las capas que hereda y las políticas de intención vigentes.

Para mover un agente a otra zona, consulta [Gestionar las Autonomy Zones](#).

Eliminar un agente

Delete agent, en el menú de acciones del agente, elimina definitivamente el agente con cada sesión y cada evento registrados para él; el historial no se puede recuperar. Si el cliente de esa máquina envía otro evento, el agente vuelve con un identificador nuevo. Retira primero el cliente de la máquina si el agente debe desaparecer del todo; consulta [Operar](#). El mismo menú exporta el agente en JSON.

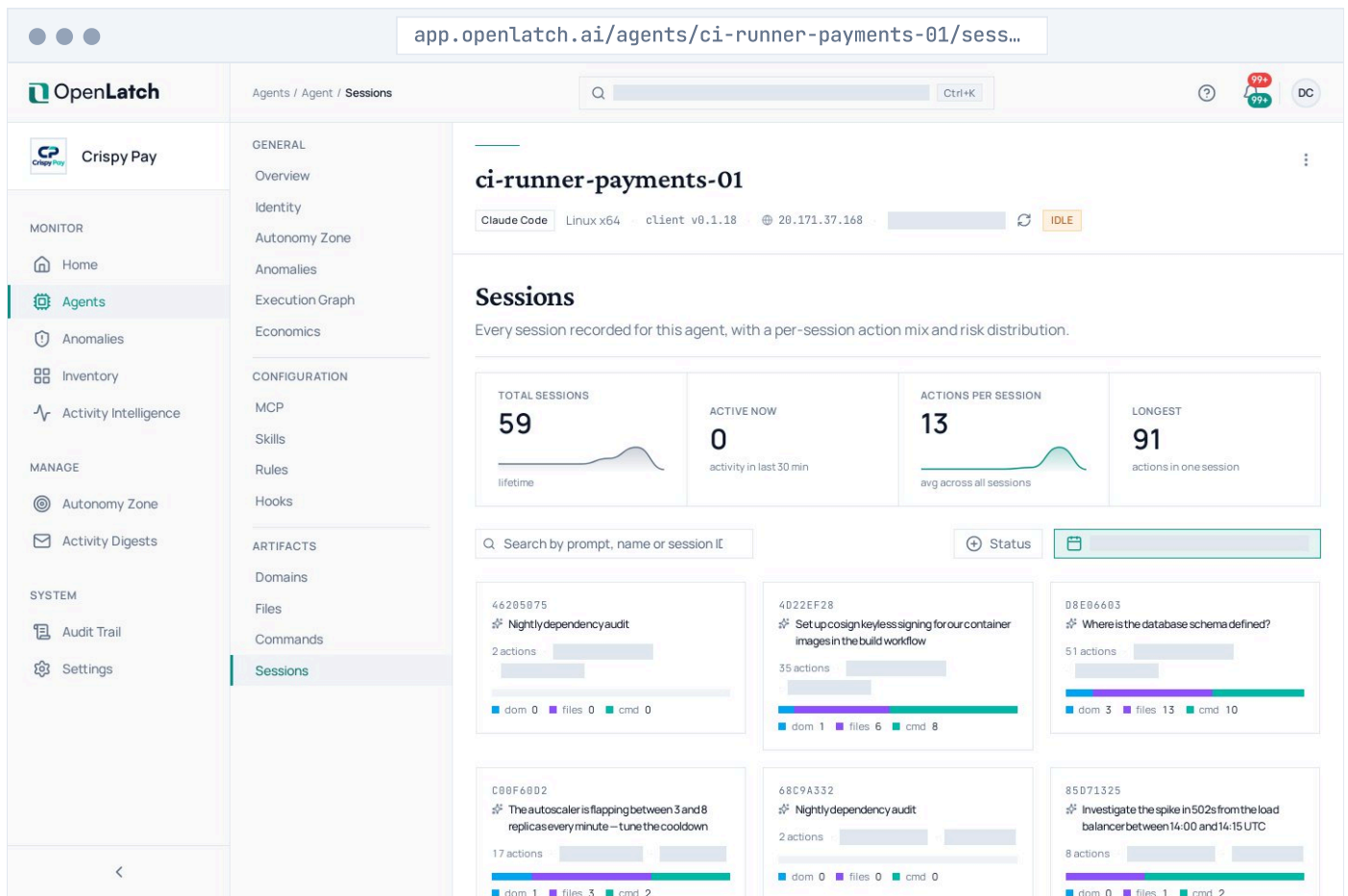
Investigar lo que hizo un agente

Usa las pestañas de actividad de un agente para responder a «¿qué hizo este agente y por qué se decidió así?». Abre el agente desde la página **Agents**, ajusta la ventana de tiempo arriba y elige la pestaña que corresponde a tu pregunta. Todos los roles pueden leerlas.

Tu pregunta	Pestaña
¿Qué hizo en una sesión de trabajo?	Sessions
¿Cómo encajan sus ejecuciones, sus herramientas y lo que tocó?	Execution graph
¿Qué comandos de shell ejecutó?	Commands
¿Qué archivos leyó, escribió o eliminó?	Files
¿A qué sitios web llegó?	Domains
¿Cuáles de sus acciones incumplieron una política de intención?	Anomalies

Sesiones

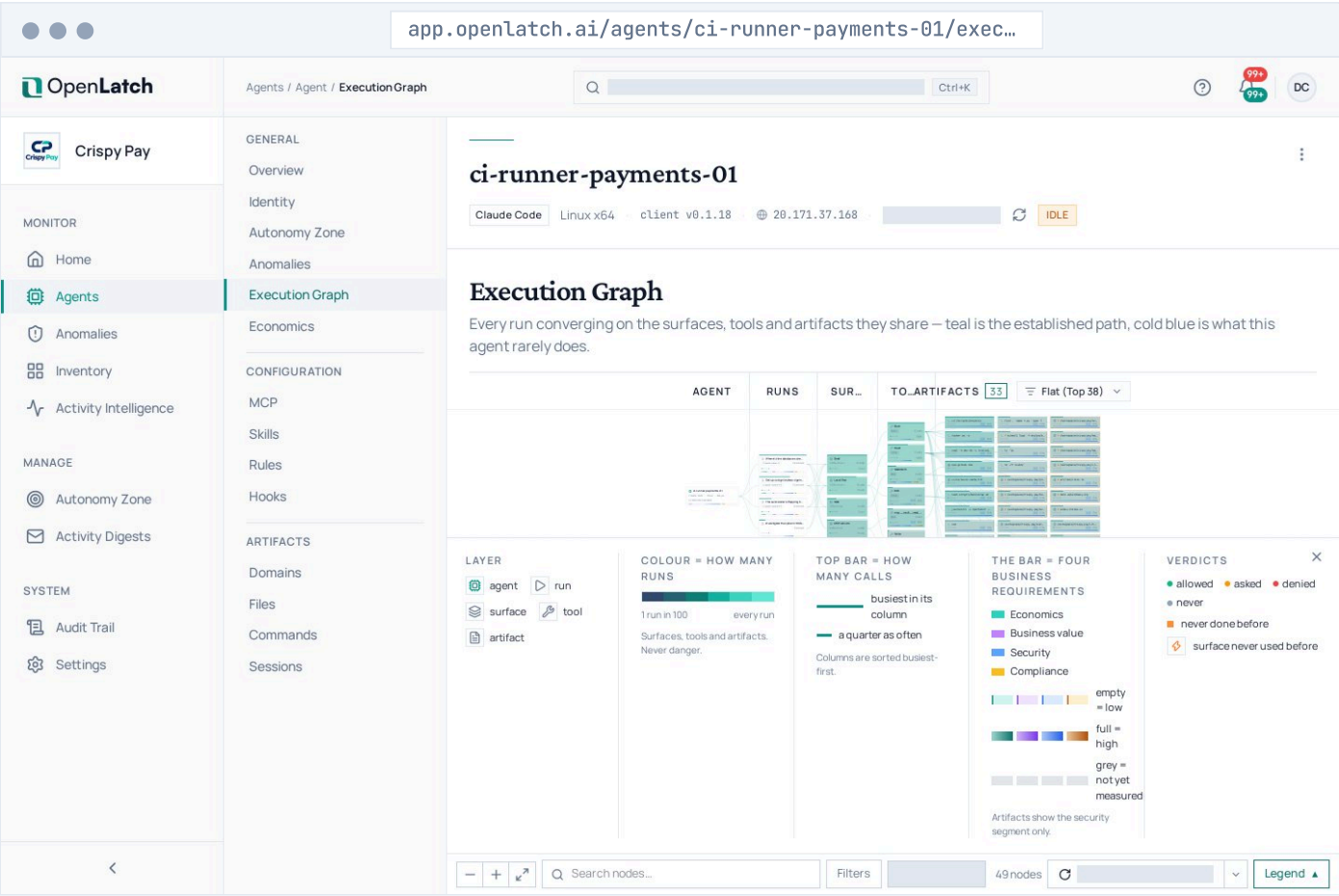
La pestaña **Sessions** enumera las sesiones de trabajo del agente, las más recientes primero. Abre una para leer sus acciones en orden, cada una con el veredicto que recibió y la regla que la decidió.



La pestaña Sessions de un agente: cada sesión de trabajo, con sus acciones en orden.

Grafo de ejecución

La pestaña **Execution graph** dibuja las ejecuciones del agente en la ventana, convergiendo en las superficies, herramientas y artefactos que comparten: el agente, luego sus ejecuciones, luego lo que buscó cada ejecución, lo que invocó, y los archivos, hosts y comandos que tocó o ejecutó. El color de un camino indica con qué frecuencia aparece en la ventana. Selecciona un nodo para leer las acciones que hay detrás. [Grafo de ejecución](#) explica el modelo.



La pestaña Execution graph de un agente: las ejecuciones convergiendo en las superficies, herramientas y artefactos que comparten.

Comandos, archivos y dominios

Estas tres pestañas enumeran cada tipo de acción del agente, con cifras de la ventana arriba. Selecciona una fila para abrir su detalle, con cada aparición y la decisión que recibió cada una.

Pestaña	Qué enumera
Commands	Cada comando de shell que ejecutó el agente, desglosado en programa y argumentos
Files	Cada archivo que el agente leyó, escribió o eliminó, con las lecturas de archivos de credenciales y las escrituras en configuración y código fuente contadas aparte
Domains	Cada sitio web al que llegó el agente, con los dominios a los que llegó por primera vez

OpenLatch

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / Commands

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1820.171.37.168IDLE

Commands

Every shell command this agent ran, ordered by time and scored per occurrence.

TOTAL COMMANDS

152

15 unique binaries

Q Search by command text...

+ Binary

TIME	BINARY	COMMAND
	cd	cd /scripts/projects
	cd	cd /src/tmp
	psql	psql -h dev-db -c truncate table scratch_events
	find	find . -name *.py -type f
	bash	bash scripts/bootstrap.sh

5 of 152 commands

<

1

2

...

7

>

La pestaña Commands de un agente: cada comando de shell que ejecutó.

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / Files

Ctrl+K

?

99+

99+

DC

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1820.171.37.168IDLE

Files

Every unique file this agent touched, classified by intent and ordered by risk.

UNIQUE PATHS5853 actions across all paths

SECRET FILES READ0credential-risk reads

CONFIGS MODIFIED4configuration writes / edits

SOURCE FILES EDITED11source-code writes / edits

Q Search by path...

Intent: All (58)

Action

Last 24 Hours

NAME	INTENT	ACTION MIX	TOUCHES	LAST SEEN
OrganizationContext.tsx /workspace/crispy-pay/webapp/src...	SOURCE		1	
biome.json /workspace/crispy-pay	CONFIG		1	
t1 kv/ci/build-cache	DOCS		12	
.gitconfig	DOCS		1	

Showing 1 to 25 of 58 files

<

1

2

3

>

La pestaña Files de un agente: cada archivo que leyó, escribió o eliminó.

app.openlatch.ai/agents/ci-runner-payments-01/doma...

OpenLatch

Agents / Agent / Domains

Crispy Pay

MONITOR

- Home
- Agents**
- Anomalies
- Inventory
- Activity Intelligence

MANAGE

- Autonomy Zone
- Activity Digests

SYSTEM

- Audit Trail
- Settings

GENERAL

- Overview
- Identity
- Autonomy Zone
- Anomalies
- Execution Graph
- Economics

CONFIGURATION

- MCP
- Skills
- Rules
- Hooks

ARTIFACTS

- Domains**
- Files
- Commands
- Sessions

ci-runner-payments-01

Claude Code Linux x64 cClient v0.1.18 20.171.37.168 IDLE

Domains

Every website this agent has touched, grouped by host and ranked by risk.

UNIQUE DOMAINS
30
37 total visits

NEW LAST 24H
2
domains not seen before by this agent

Q Search domain, country... All (30) New last 24h (2)

DOMAIN	COUNTRY - ASN	IPS	TOUCHES	LAST SEEN	TREND
index.crates.io	UNK	151.101.46.137	1		
api.github.com	UNK	140.82.112.6 +3	7		
docs.sqlalchemy.org	UNK	104.26.11.194	1		
api.cohere.ai	UNK	34.96.76.122	1		
cloud.google.com	UNK	172.253.63.100	1		

Showing 1 to 25 of 30 domains

La pestaña Domains de un agente: cada sitio web al que llegó.

El texto de los comandos y las rutas de archivo se enmascaran en la máquina antes de enviarse, así que una credencial nunca aparece aquí. Consulta [Tratamiento de datos y seguridad](#).

Anomalías

La pestaña **Anomalies** solo enumera las anomalías de este agente: sus acciones que incumplieron una política de intención de su zona, cada una con la política, su dimensión y el veredicto. [Revisar las anomalías](#) explica cómo trabajarlas en toda la organización.

app.openlatch.ai/agents/ci-runner-payments-01/anom...

OpenLatch

Agents / Agent / **Anomalies**

Crispy Pay

MONITOR

- Home
- Agents**
- Anomalies
- Inventory
- Activity Intelligence

MANAGE

- Autonomy Zone
- Activity Digests

SYSTEM

- Audit Trail
- Settings

GENERAL

- Overview
- Identity
- Autonomy Zone
- Anomalies**
- Execution Graph
- Economics

CONFIGURATION

- MCP
- Skills
- Rules
- Hooks

ARTIFACTS

- Domains
- Files
- Commands
- Sessions

ci-runner-payments-01

Claude Code Linux x64 cClient v0.1.18 IDLE

Anomalies

Intent-policy violations raised by this agent.

Waiting on your answer All

WAITING ON YOUR ANSWER 0 nothing waiting	ANOMALIES - 7 DAYS 0 across 0 agents	BLOCKED 0	ALLOWED AND FLAGGED 0
--	--	--------------	--------------------------

Nothing is waiting on your answer
Every flagged action has an answer against it, so the policy knows what to do next time.

La pestaña Anomalies de un agente: sus acciones que incumplieron una política de intención.

Convertir una observación en una política

Cuando una acción muestra algo que quieres que se decida de otra forma la próxima vez, redacta la política de intención que lo diga: consulta [Redactar una Agent Intent Policy](#). OpenLatch solo genera una anomalía donde una política ratificada cubre la acción.

Revisar la configuración de un agente

Usa las pestañas de configuración de un agente para ver qué está configurado para hacer en su máquina, antes de que lo haga: los hooks que declara, los servidores MCP a los que se conecta, los archivos de reglas que lee y los skills que puede ejecutar. El cliente envía esta configuración desde la máquina, así que un cambio hecho allí aparece aquí. Todos los roles pueden leer estas pestañas.

Abre el agente desde la página **Agents** y elige una pestaña.

Pestaña	Qué enumera	Cifras de arriba
Hooks	Los hooks de ciclo de vida que declara el agente, incluidos los que lo conectan al cliente	Bloques de hooks, sus manejadores y las fuentes modificadas
MCP servers	Los servidores conectados al host MCP del agente, con las herramientas que expone cada uno	Servidores, herramientas en total y fuentes modificadas
Rules	Los archivos de reglas y la memoria del agente encontrados en el host	Archivos, fuentes modificadas y el archivo más grande
Skills	Cada skill y comando de barra que el agente puede ejecutar	Skills y fuentes modificadas

Selecciona una entrada para abrir su detalle: dónde se declaró, su contenido y cuándo cambió por última vez.

OpenLatch

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / Hooks

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.18IDLE

Hooks

Lifecycle hooks declared by this agent.

TOTAL HOOK BLOCKS

4

Across this host

HANDLERS

7

Across all blocks

MODIFIED 7D

4

Sources mutated

Q Search lifecycle hooks...

LifecycleLast change

NAME	LIFECYCLE	HANDLERS	LAST CHANGED
Claude Code hooks	PreToolUsePostToolUse+1	3	
Claude Code hooks	PostToolUse	1	
Claude Code hooks	PostToolUse	1	
Claude Code hooks	StopSessionStart	2	

Showing 1 to 4 of 4 sources

La pestaña Hooks de un agente: los hooks de ciclo de vida que declara.

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Agents / Agent / MCP

Q

Ctrl+K

?

99+

99+

DC

GENERAL

Overview

Identity

Autonomy Zone

Anomalies

Execution Graph

Economics

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

ci-runner-payments-01

Claude CodeLinux x64cClient v0.1.1820.171.37.168IDLE

MCP servers

Servers connected to this agent's MCP host.

TOTAL SERVERS

12

Across this host

TOOLS REGISTERED

19

Across all servers

CHANGED 7D

12

Sources mutated

Q Search servers...

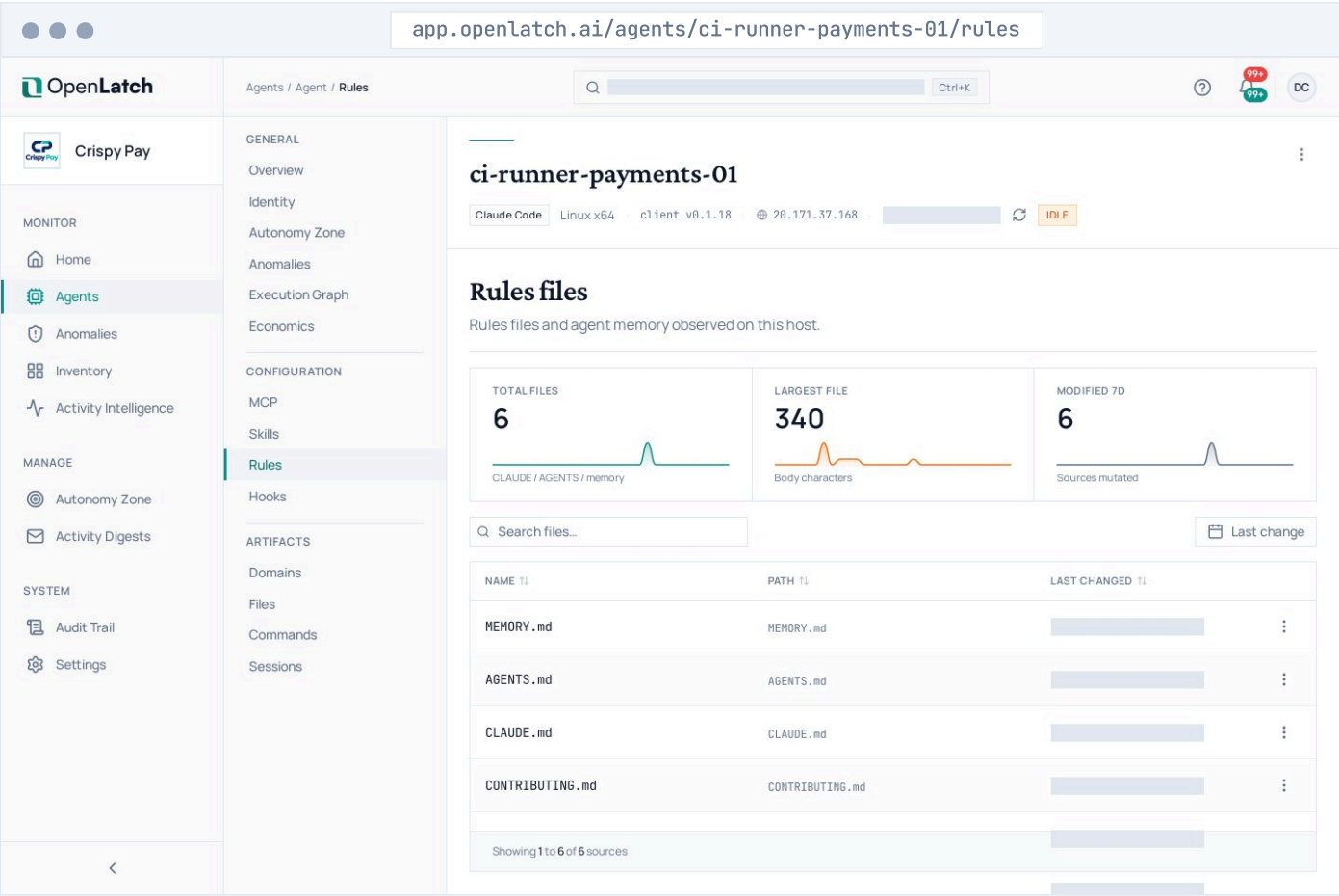
Scope: AllEnterprisePersonalProjectLocal

Last change

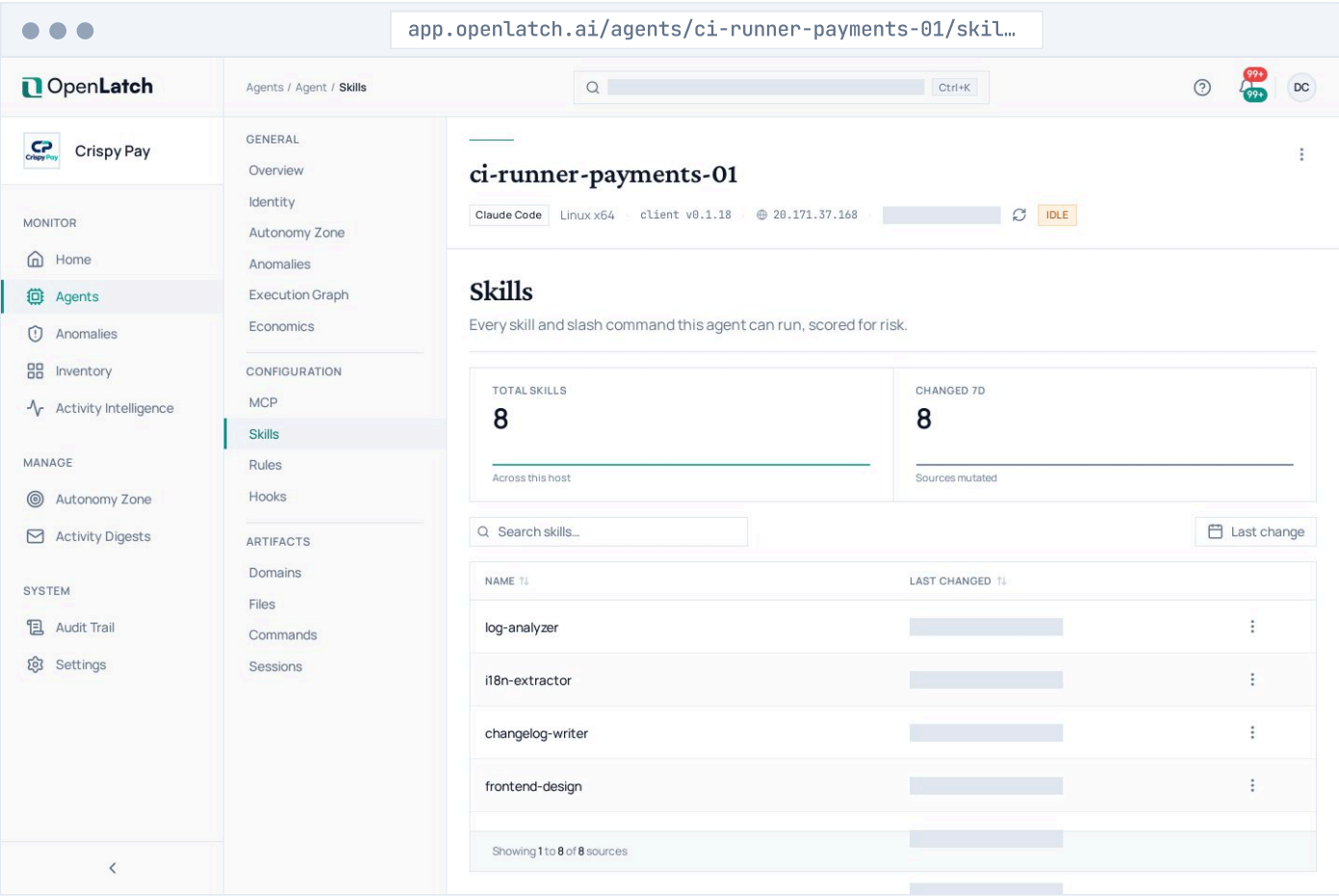
NAME	SCOPE	TOOLS	LAST CHANGED
mongodb	project	1	
postgres	project	2	
datadog	project	1	
twilio	personal	1	

Showing 1 to 12 of 12 sources

La pestaña MCP servers de un agente: los servidores conectados a su host MCP y sus herramientas.



La pestaña Rules de un agente: los archivos de reglas y la memoria encontrados en su host.



La pestaña Skills de un agente: cada skill y comando de barra que puede ejecutar.

Qué mirar

Comprobación

El hook que conecta el agente al cliente está presente

Un servidor MCP o un skill inesperado

Una fuente modificada recientemente

Por qué

Sin él, las acciones del agente no llegan al punto de decisión. Ejecuta `openlatch doctor` en la máquina si falta

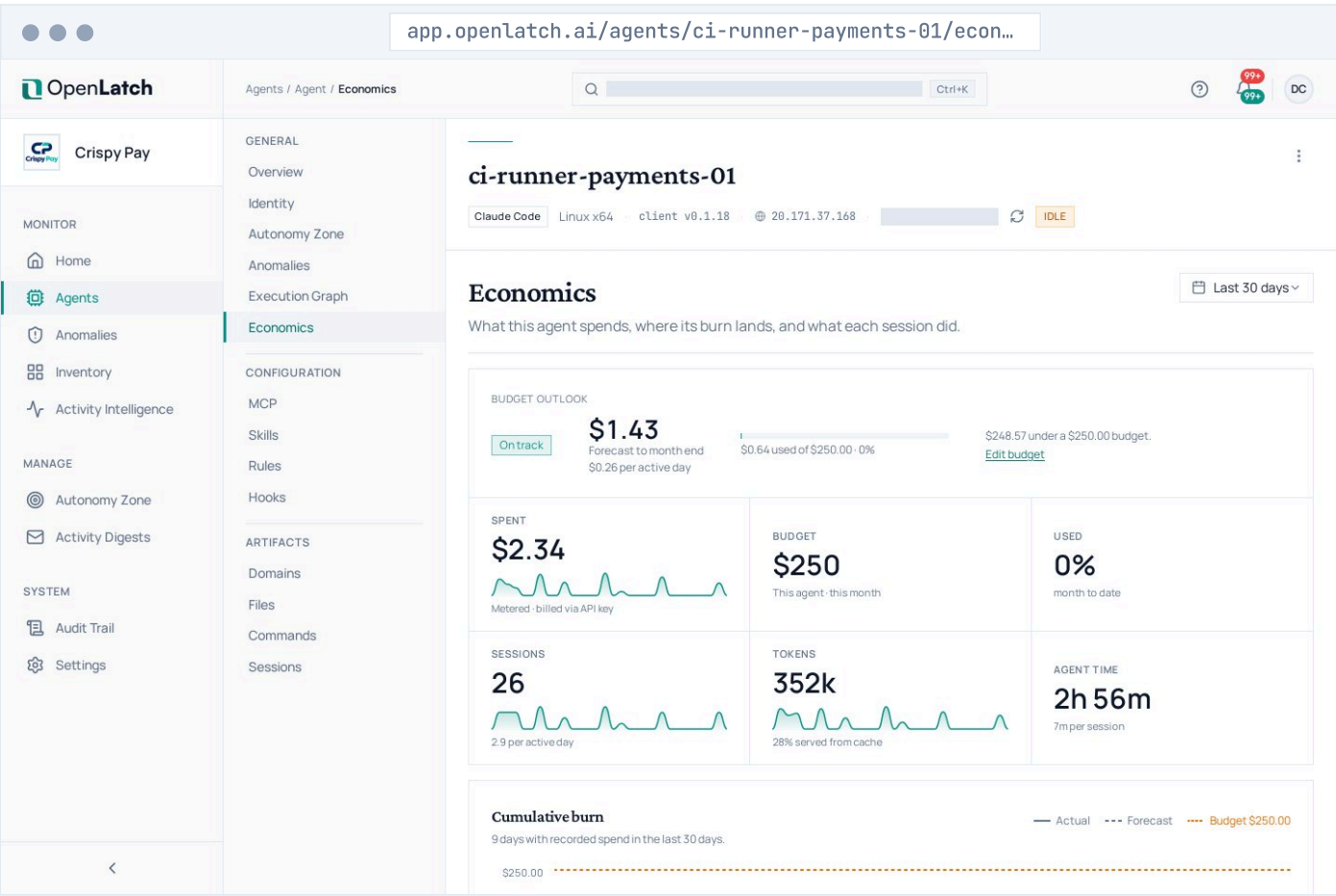
Un servidor o skill nuevo da al agente nuevas herramientas con las que actuar. Redacta una política de intención si su uso debe decidirse

Un cambio en un archivo de reglas o en un hook cambia el comportamiento del agente desde su siguiente sesión

Una pestaña vacía significa que el agente aún no ha declarado nada de ese tipo. Para ver la misma configuración en todos los agentes a la vez, usa el [inventario](#).

Seguir el gasto de un agente

Usa la pestaña **Economics** de un agente para ver lo que gasta en modelos, dónde se concentra ese gasto y lo que produjo cada sesión. Todos los roles pueden leerla; definir un presupuesto requiere el rol admin o superior.



La pestaña Economics de un agente: gasto, previsión del presupuesto y coste de cada sesión.

Leer el gasto

Sección	Qué muestra
Gasto	Lo que gastó el agente en la ventana. El gasto facturado con una clave de API es medido; el gasto con una suscripción es estimado, y se indica así
Previsión del presupuesto	Cómo se compara el gasto del mes con el presupuesto del agente
Consumo acumulado	El gasto real hasta la fecha, proyectado hasta el final del periodo
Coste por modelo	Cómo se reparte el gasto entre los modelos que usó el agente
Parte de la entrada servida desde caché	Qué parte de la entrada del agente sirvió el proveedor de modelos desde su caché
Sesiones	Cada sesión con su coste, su duración y las herramientas que usó. Una sesión aún en curso muestra un total provisional, no una cifra final

Las peticiones y respuestas de modelos nunca se envían a OpenLatch; el gasto se obtiene del número de tokens, el modelo usado y el reparto de caché que registra el cliente. Consulta [Tratamiento de datos y seguridad](#). Cuando se desconoce el precio de un modelo, la pestaña indica que falta parte del gasto en sus cifras.

Definir un presupuesto

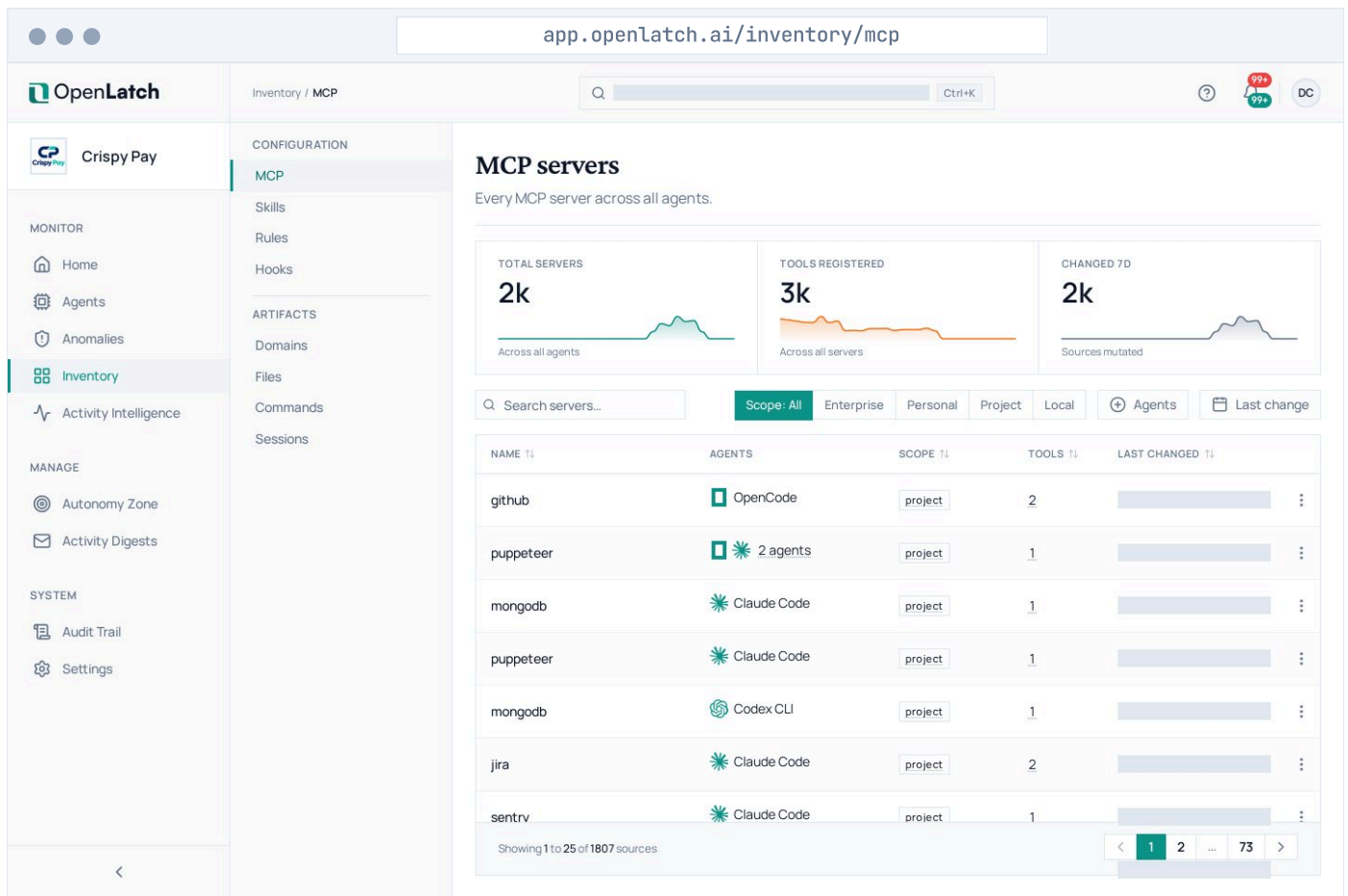
- 1 En la pestaña **Economics**, abre el presupuesto mensual del agente.
- 2 Introduce el importe del mes y guárdalo. El presupuesto se reinicia el día uno de cada mes; no se acumula nada.

Un agente sin presupuesto propio usa el presupuesto mensual por defecto de tu organización, definido en **Settings > Organization Profile**. Ese valor por defecto se aplica a cada uno de esos agentes por separado: un valor por defecto de 500 \$ da a cada agente 500 \$ al mes, no 500 \$ compartidos. Consulta [Editar el perfil de la organización](#).

Buscar en el inventario

Usa la página **Inventory** cuando tu pregunta empieza por una cosa y no por un agente: qué agentes se conectan a este servidor MCP, cuáles ejecutaron este comando, cuáles llegaron a este dominio. Cada pestaña enumera un tipo de cosa en todos los agentes de la organización, contada una sola vez por muchos agentes que la compartan. Todos los roles pueden leerla.

Pestaña	Enumera, en todos los agentes
Commands	Cada comando de shell que ejecutaron los agentes
Files	Cada archivo que los agentes leyeron, escribieron o eliminaron
Domains	Cada sitio web al que llegaron los agentes
Sessions	Cada sesión de trabajo, con el agente al que pertenece
Hooks	Cada bloque de configuración de hooks que declaran los agentes
MCP servers	Cada servidor MCP al que se conectan los agentes
Rules	Cada archivo de reglas que leen los agentes
Skills	Cada skill que pueden ejecutar los agentes

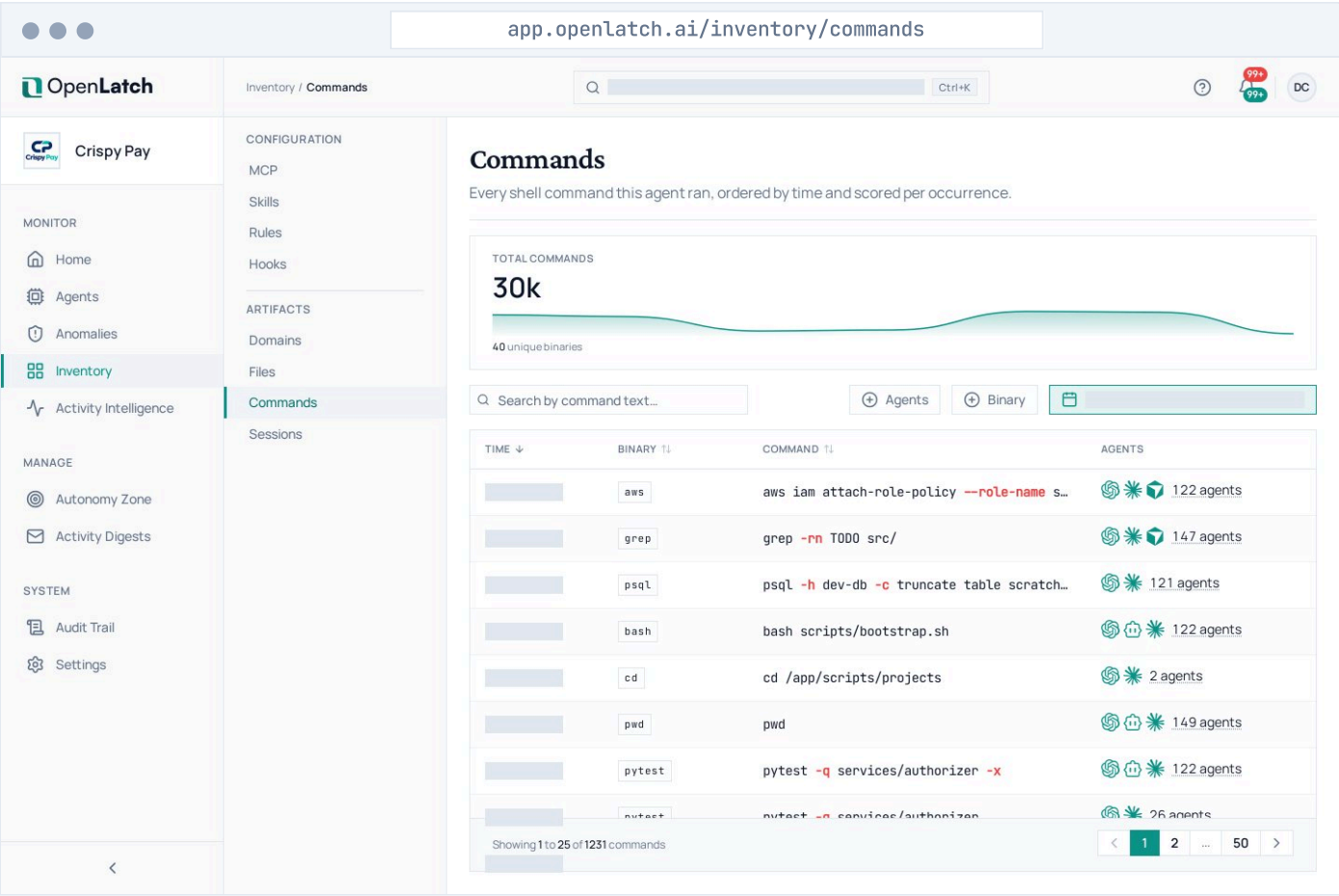


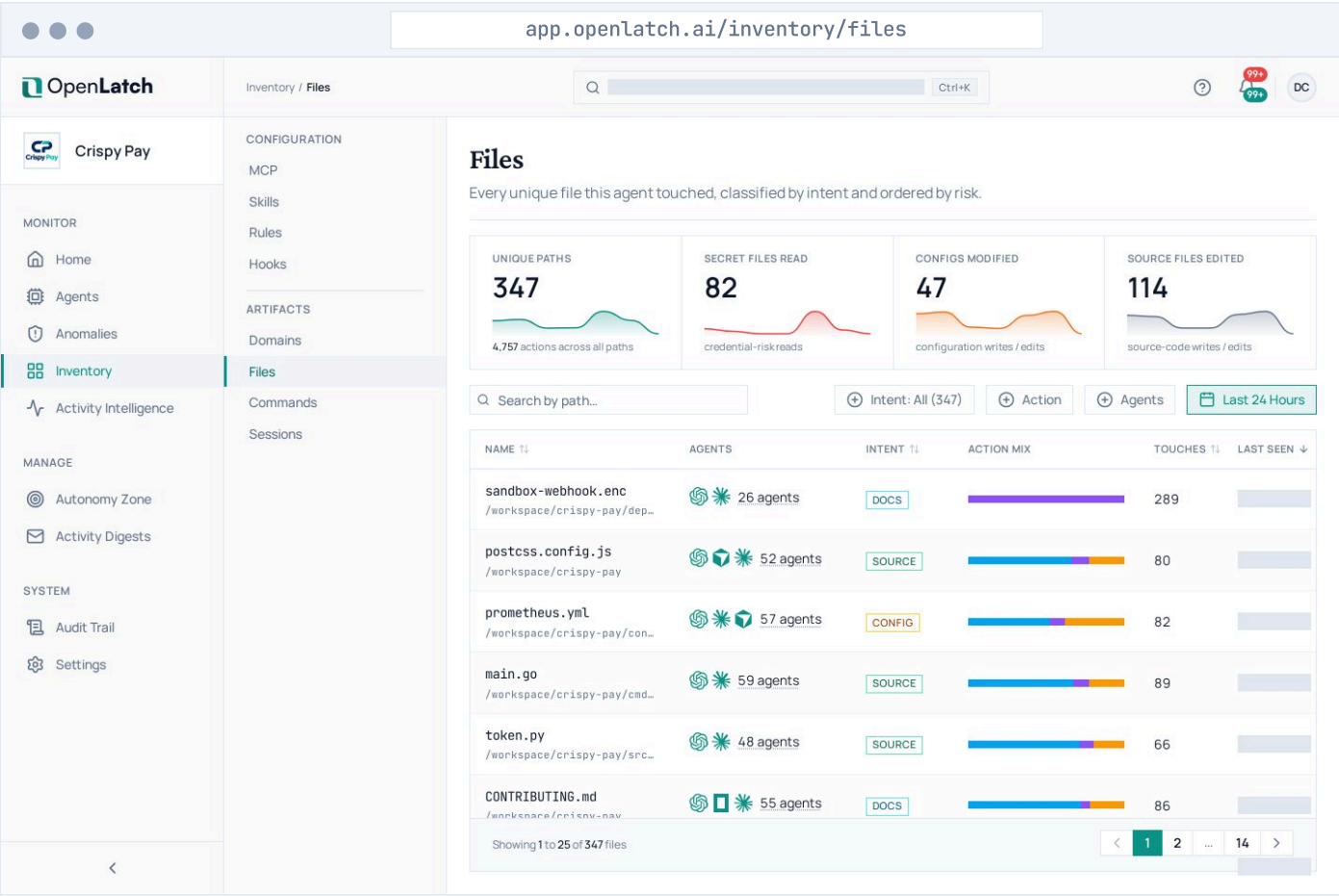
El inventario MCP servers: cada servidor, con el número de agentes en los que se vio.

Encontrar qué agentes usan algo

1. Abre **Inventory** y la pestaña del tipo de cosa que buscas.
2. Búscala, o filtra la lista.
3. Lee la columna **Agents**. Pasa el cursor por encima para ver en qué agentes se vio la entrada, o abre la entrada para ver cada aparición, cada una enlazada con su agente.

Desde una aparición, abre el agente para leer la acción en su contexto; consulta [Investigar lo que hizo un agente](#).





OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Inventory / Domains

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

app.openlatch.ai/inventory/domains

Q

Ctrl+K

?

99+

99+

DC

Domains

Every website this agent has touched, grouped by host and ranked by risk.

UNIQUE DOMAINS

81

2,889 total visits

NEW LAST 24 H

0

domains not seen before by this agent

Q Search domain, country...

All (81)

New last 24h (0)

+ Agents

DOMAIN	AGENTS	COUNTRY - ASN	IPS	TOUCHES	LAST SEEN
cloud.google.com	43 agents	UNK	172.253.63.138 +4	66	
react.dev	41 agents	UNK	66.33.60.34 +7	61	
status.crisppay.com	43 agents	UNK	—	67	
docs.rs	50 agents	UNK	151.101.1.91 +3	67	
playwright.dev	46 agents	UNK	185.199.109.153 +2	54	
stackoverflow.com	50 agents	UNK	198.252.206.1	68	
go.dev	48 agents	UNK	216.239.38.21 +7	66	

Showing 1 to 25 of 81 domains

<

1

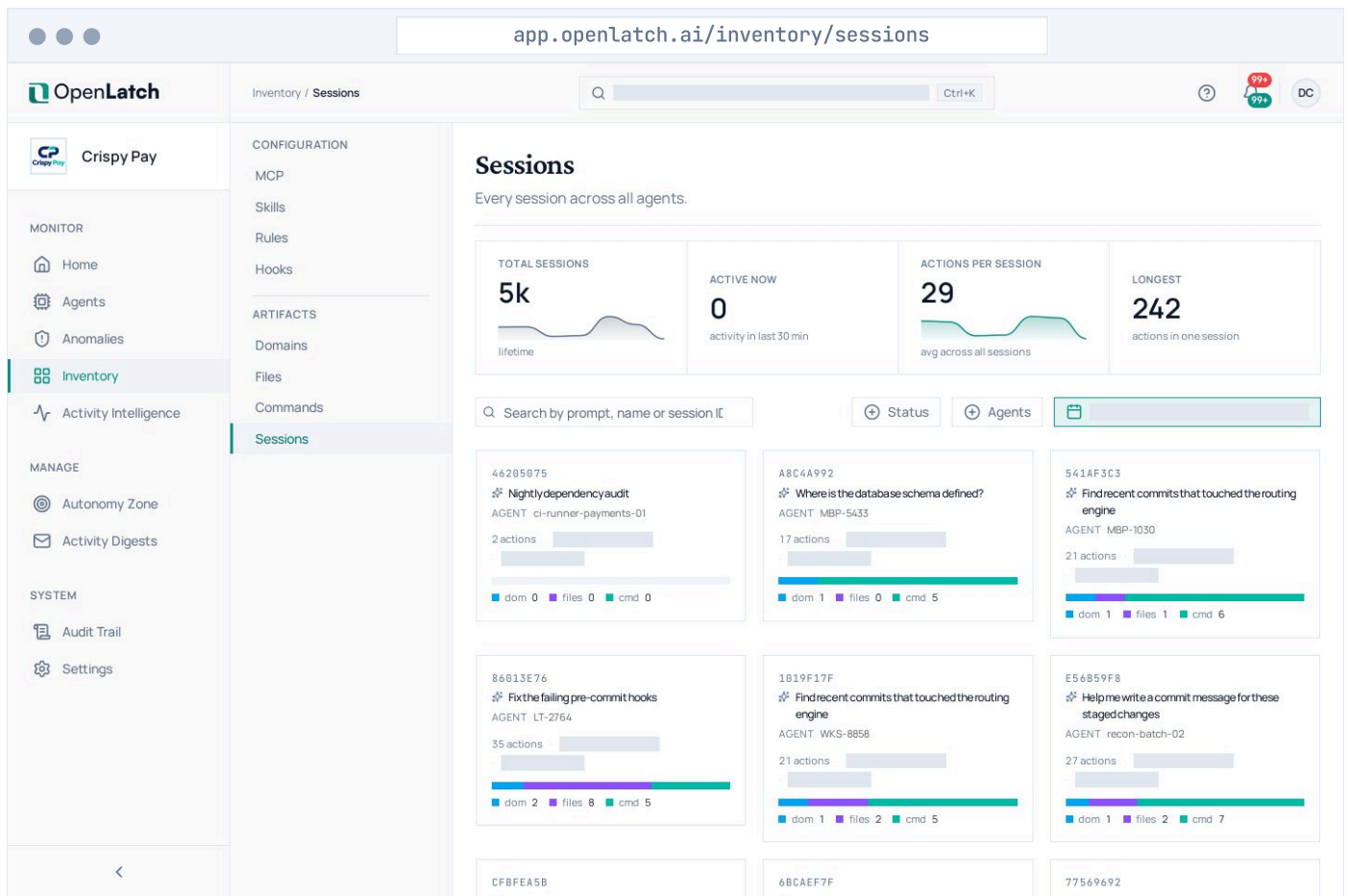
2

3

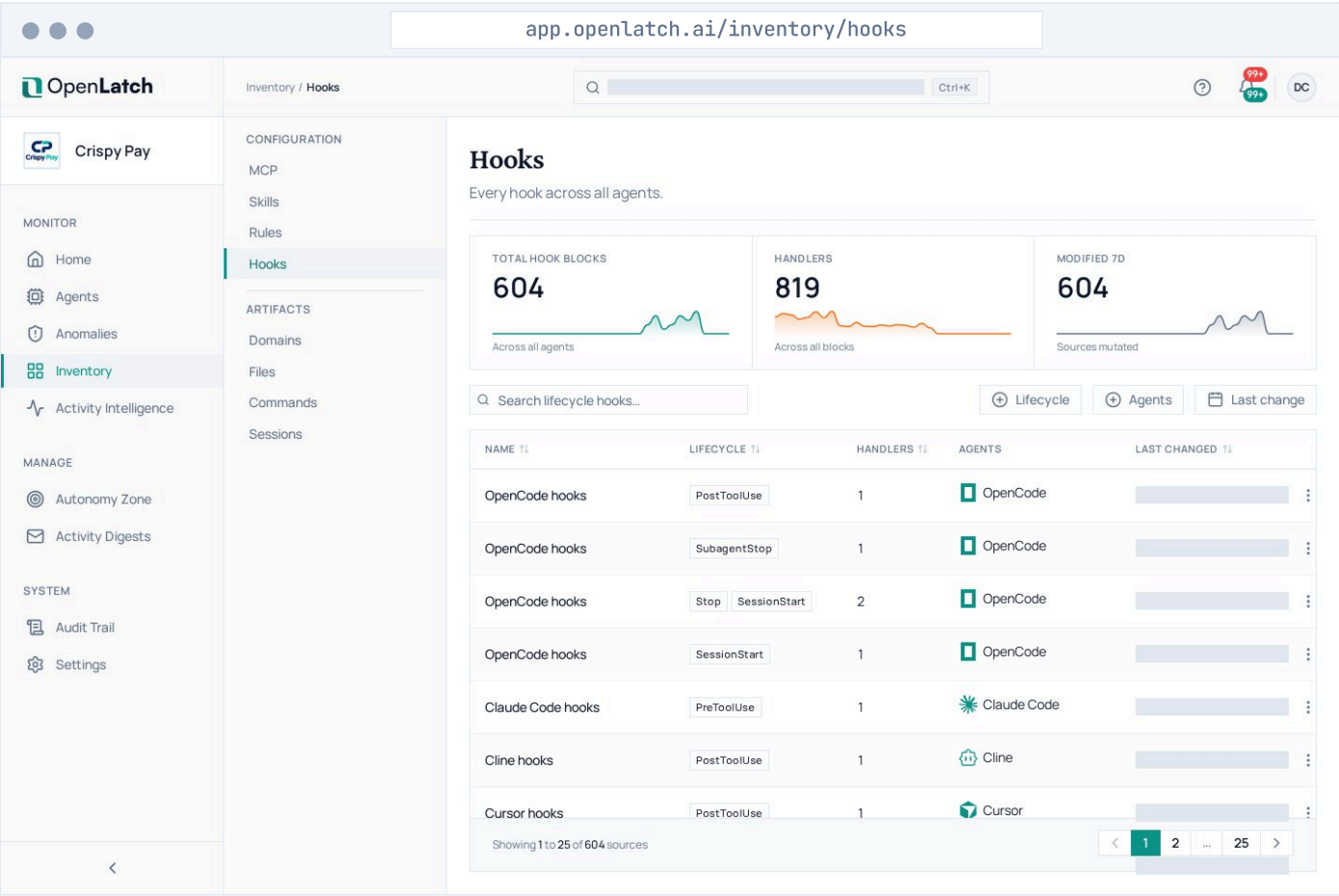
4

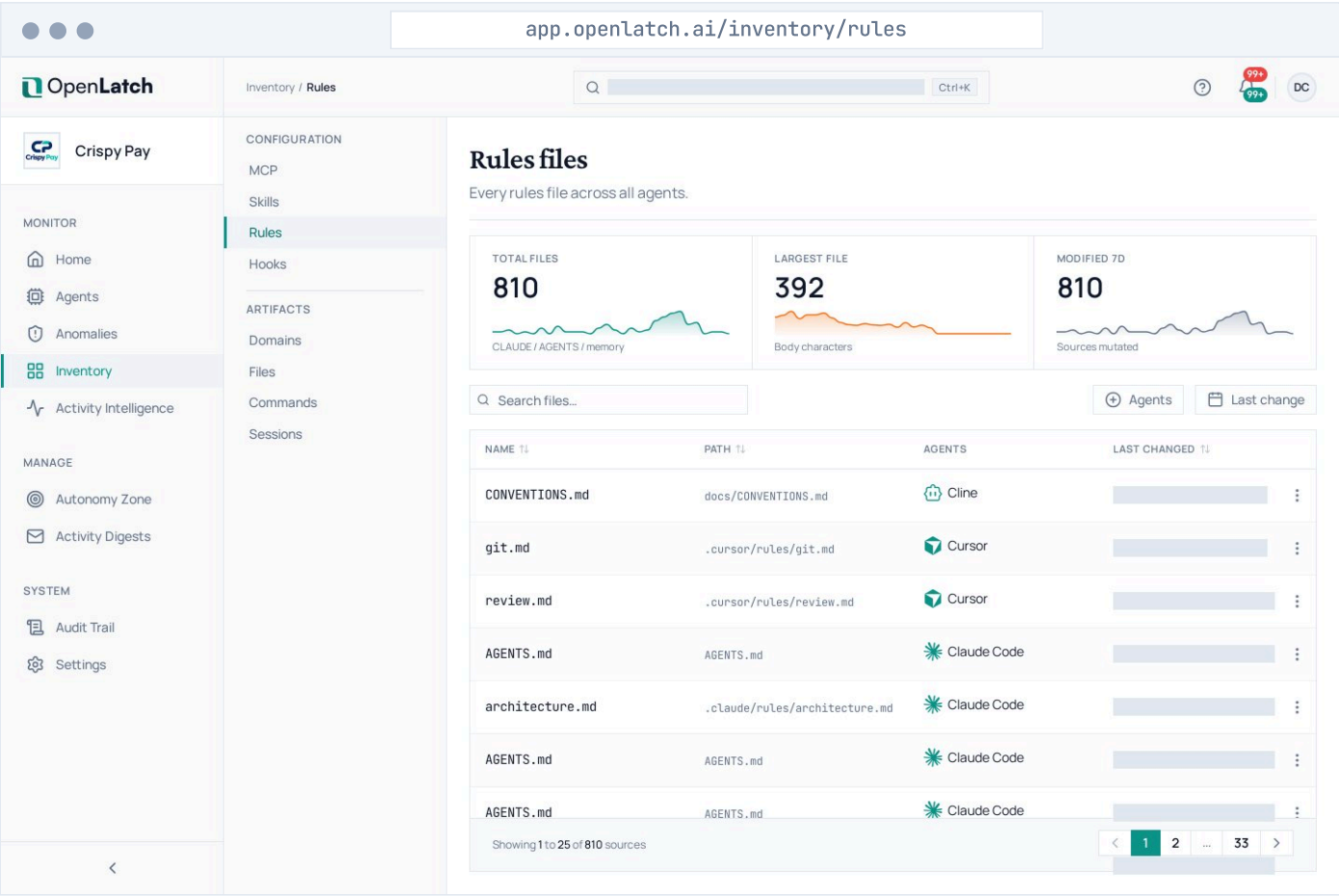
>

El inventario Domains: cada sitio web al que llegaron los agentes.



El inventario Sessions: cada sesión de trabajo, con el agente al que pertenece.





app.openlatch.ai/inventory/skills

OpenLatch

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Inventory / Skills

CONFIGURATION

MCP

Skills

Rules

Hooks

ARTIFACTS

Domains

Files

Commands

Sessions

Skills

Every skill across all agents.

TOTAL SKILLS

720

Across all agents

CHANGED 7D

720

Sources mutated

Search skills...

Agents

Last change

NAME ↕	AGENTS	LAST CHANGED ↕
ci-pipeline	Claude Code	⋮
refactor-helper	Claude Code	⋮
migration-author	Claude Code	⋮
refactor-helper	Claude Code	⋮
test-author	Claude Code	⋮
log-analyzer	Claude Code	⋮
log-analyzer	Claude Code	⋮

Showing 1 to 25 of 720 sources

<

1

2

...

29

>

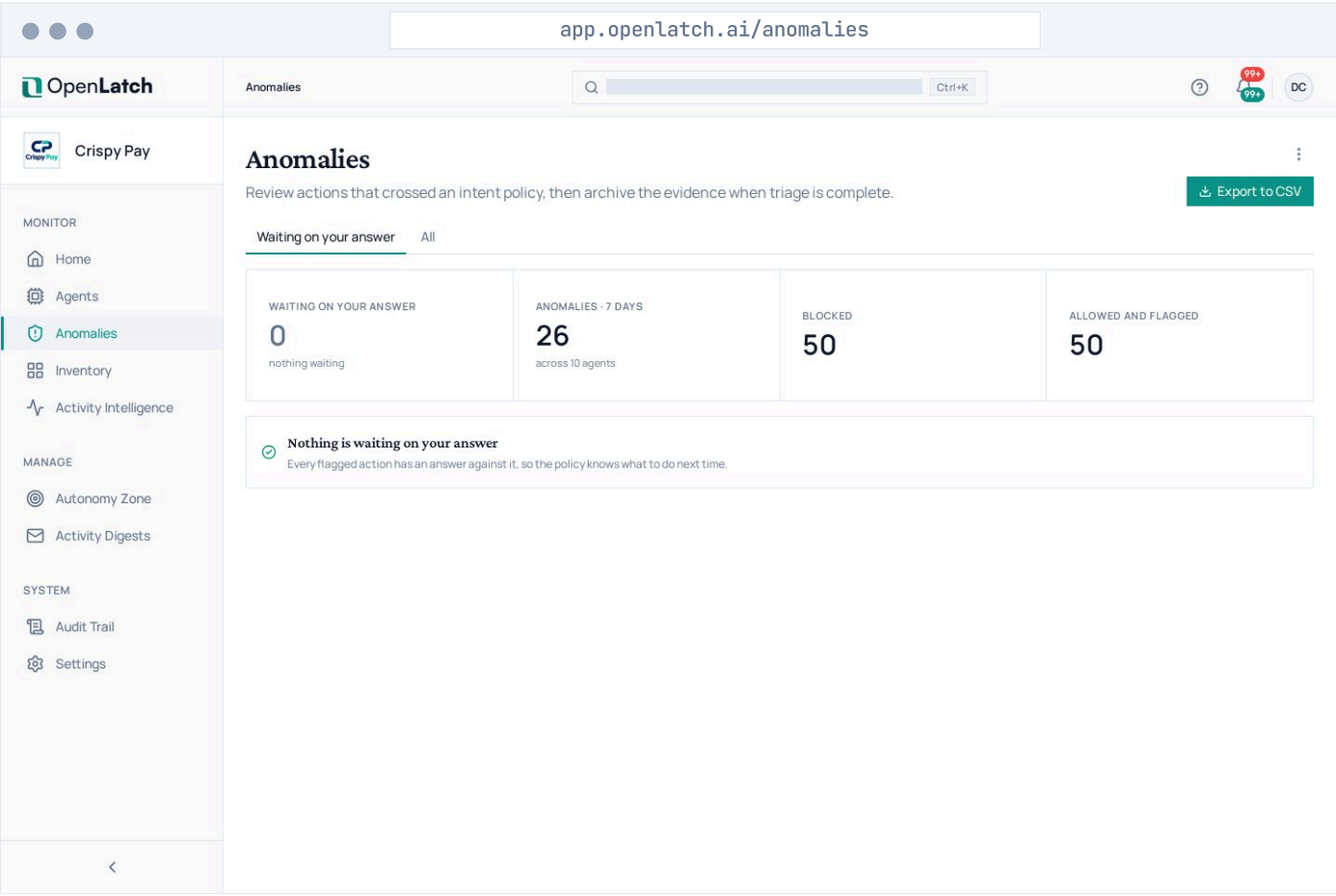
El inventario Skills: cada skill que pueden ejecutar los agentes.

Una pestaña vacía significa que ningún agente ha declarado ni hecho nada de ese tipo todavía.

Revisar las anomalías

Usa la página **Anomalies** para trabajar cada acción de agente de tu organización que incumplió una política de intención. Todos los roles pueden leerla; responder a un Ask requiere el rol editor o superior.

Una anomalía es una acción de agente que incumple una política de intención de una de las dimensiones de la Autonomy Zone del agente. Cada una nombra la política, su dimensión, la zona de la que procede la política y el veredicto que recibió la acción. [Anomalías](#) explica el modelo.



La página Anomalies: cada acción que incumplió una política de intención, con la política, su dimensión y el veredicto.

Trabajar la lista

1. Ajusta la ventana de tiempo y usa los filtros para acotar la lista a lo que revisas.

2. Abre una anomalía. Su detalle muestra de qué Autonomy Zone y de qué política procede, qué intención coincidió, la acción que la generó, con qué frecuencia ocurre y el agente.
3. Abre el agente desde la anomalía para leer la acción en su sesión; consulta [Investigar lo que hizo un agente](#).

Lo que significa una anomalía depende del modo de la política:

Modo de la política	Qué le pasó a la acción
Monitor	Continuó, y la anomalía registra lo que la política habría decidido
Enforce	Recibió el veredicto de la política antes de ejecutarse, y la anomalía lo registra

Responder a un Ask

Cuando la ASK delivery de una zona incluye la plataforma, las acciones marcadas por un Ask aparecen en la parte superior de esta página para que una persona responda. Por defecto, esas acciones ya se ejecutaron: tu respuesta no las deshace, decide lo que la política aplicará la próxima vez que encuentre una acción similar. Ask evita la repetición, no la primera vez.

Respuesta	Efecto
Allow next time	La política deja que las acciones similares se ejecuten a partir de ahora
Block next time	La política detiene las acciones similares a partir de ahora
Not now – 24 h	La pregunta vuelve en un día. Hasta que respondas, se vuelve a disparar por el mismo motivo

Cuando una política activó, por átomo, que una acción irreversible se detenga hasta que alguien responda, la acción figura como bloqueada hasta la respuesta. Cada respuesta queda en el registro de auditoría con tu nombre. [Gestionar las Autonomy Zones](#) cubre la ASK delivery.

Desde el detalle de una anomalía, **Promote to Enforce** pasa su política a Enforce en un paso, cuando la política tiene intenciones aplicables.

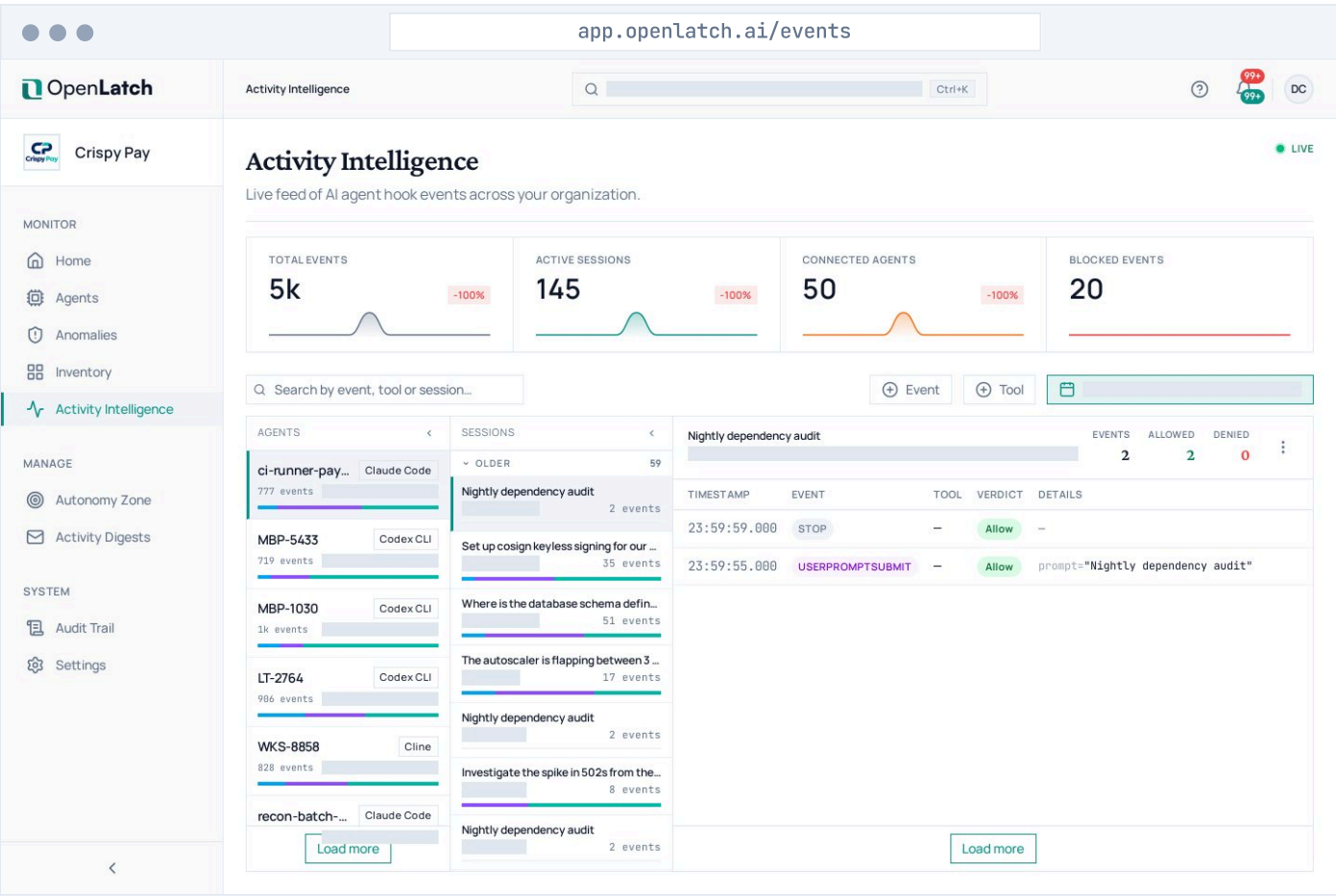
Ajustar la política detrás de una anomalía

Una anomalía en un trabajo que consideras normal significa que la política dice más de lo que querías. Abre la política desde la anomalía, responde No a su frase diciendo lo que querías decir y ratifica la nueva versión; consulta [Redactar una Agent Intent Policy](#). Una política cuyas anomalías son todas acciones sobre las que quieres actuar está lista para [pasar a Enforce](#).

Para ver solo las anomalías de un agente, abre su pestaña **Anomalies** desde la página **Agents**.

Seguir los eventos en directo

Usa **Activity Intelligence**, la página de eventos en directo de la consola en /events, para ver lo que hacen los agentes ahora mismo: cada evento de hook de tu organización, en cuanto llega. Todos los roles pueden leerla.



Activity Intelligence: el flujo en directo de eventos de hooks de los agentes, agrupados por sesión.

Leer el flujo

Las cifras de arriba cuentan, en el intervalo de tiempo seleccionado, los eventos recibidos, las sesiones activas, los agentes conectados y los eventos bloqueados, cada uno comparado con la mitad anterior de la ventana. Un indicador en directo muestra que el flujo está conectado.

Debajo, las sesiones aparecen a la izquierda, agrupadas en **Active now**, **Recent** y **Older**. Selecciona una sesión para leer sus eventos a la derecha, en orden. Abre un evento para ver su detalle.

Para	Haz esto
Encontrar un evento	Busca por evento, herramienta o sesión
Acotar el flujo	Filtra por tipo de evento, herramienta o intervalo de tiempo
Empezar de cero	Elige Clear filters

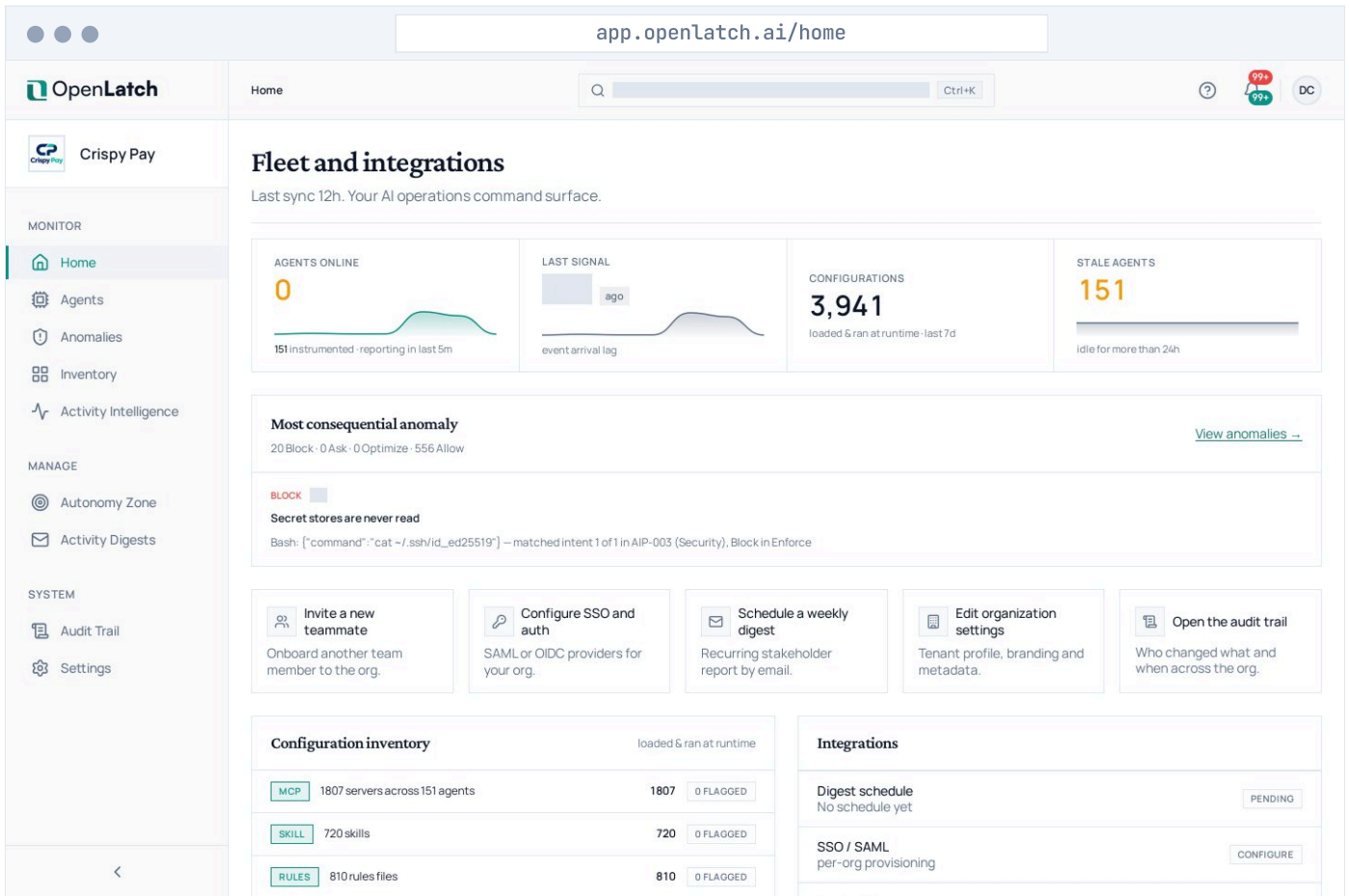
Antes del primer evento

Justo después de conectar un cliente, la página muestra **Awaiting first agent activity**: el cliente está conectado y aún no ha ocurrido nada. El primer evento aparece en cuanto un agente invoca una herramienta. Si no aparece nada después de que el agente haya trabajado un rato, ejecuta `openLatch doctor` en la máquina; consulta [Verificar y desplegar](#).

Para el historial completo de un agente en lugar del flujo en directo, usa sus pestañas; consulta [Investigar lo que hizo un agente](#).

Leer la página de inicio

La página **Home** es la que abre la consola después de iniciar sesión. Resume el estado de tu organización y señala lo que merece una mirada. Abrir `app.openlatch.ai` sin ruta te trae aquí.



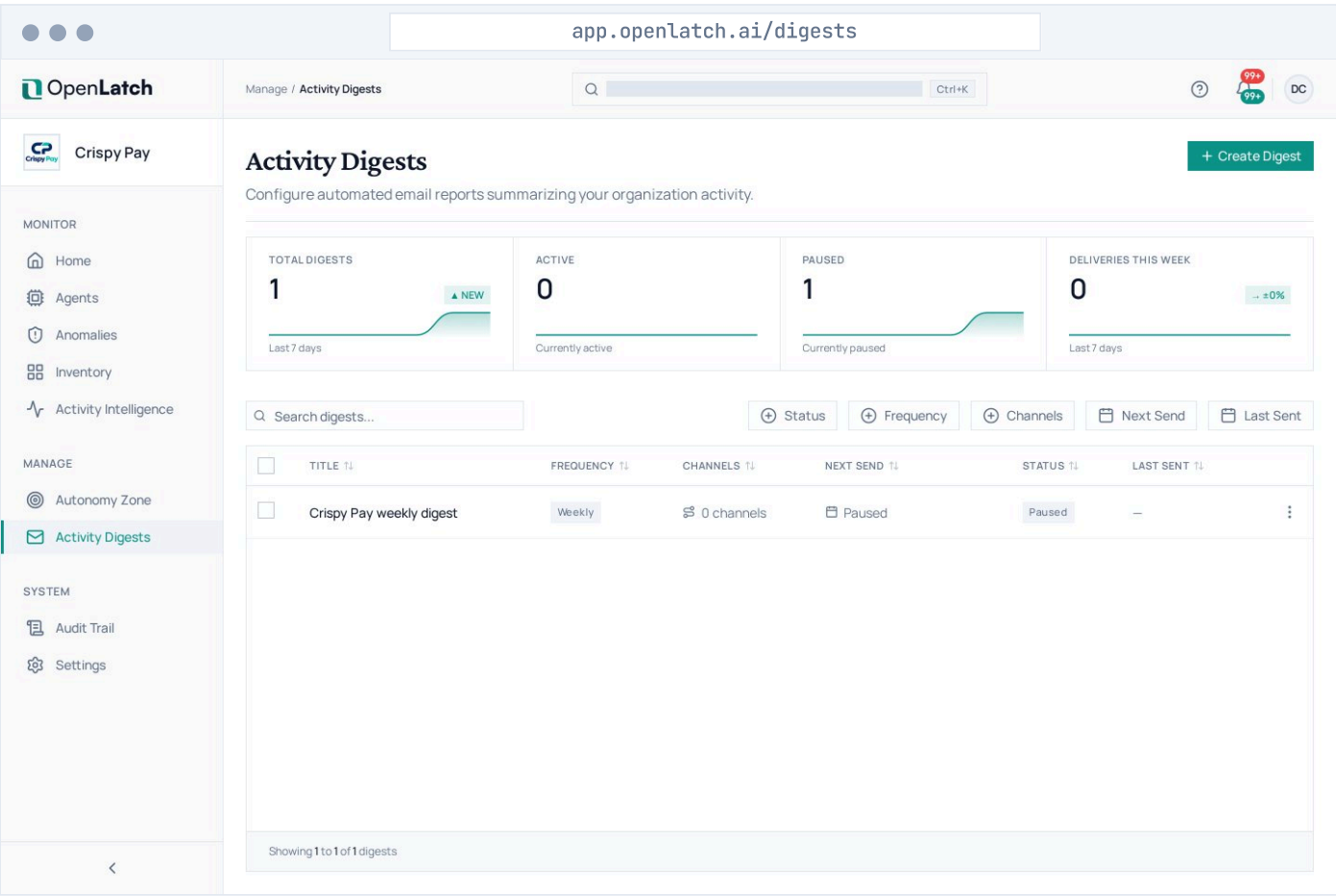
La página Home: los agentes que envían datos, lo que cambió desde tu última visita y lo siguiente que conviene mirar.

Panel	Qué te indica	Adónde lleva
Fleet health	Cuántos de tus agentes envían datos	Encontrar y leer un agente
Configuration inventory	Cuántas fuentes de configuración (hooks, servidores MCP, archivos de reglas, skills) se añadieron, modificaron o eliminaron desde tu última visita	Buscar en el inventario
Integrations	Tus programaciones de resúmenes activas	Programar resúmenes de actividad
Single sign-on	El estado de tus métodos de inicio de sesión. Leerlo requiere un rol de administrador	Configurar el inicio de sesión único
Recent changes	Las últimas entradas del registro de auditoría	Revisar el registro de auditoría
Team	Cuántos miembros tiene la organización	Gestionar usuarios y equipos
Most active users	Las personas cuyos agentes hicieron más acciones en los últimos siete días	Gestionar usuarios y equipos
If you have 5 minutes	Una breve lista de lo que hacer a continuación, como revisar una fuente de configuración modificada, conectar otro agente o configurar un resumen, o un aviso de que todo está al día	La página de cada paso

La lista **Since you last visited** nombra cada cambio de configuración desde tu visita anterior. Abre uno para leerlo en el inventario. **Reopen setup** te devuelve al asistente de configuración de la organización.

Programar resúmenes de actividad

Usa **Digests** para enviar por correo, según una programación, un informe sobre la actividad de los agentes de tu organización, a personas con o sin cuenta en la consola. Necesitas el rol editor o superior para crear o cambiar un resumen.



Digests: cada programación con su frecuencia, sus canales, su próximo envío y su estado.

Crear un resumen

- 1 Abre **Digests** y elige **Create Digest**.
- 2 Dale un título y un asunto. El asunto puede usar {period} y {organization_name}.
- 3 En **Notification Routing**, selecciona tu canal Email. El campo de destinatarios solo aparece cuando hay un canal Email seleccionado.

- 4 Añade los destinatarios, hasta 100 direcciones de correo. Los destinatarios no necesitan una cuenta en la consola.
- 5 Elige la frecuencia, diaria, semanal, mensual o trimestral, y después el día, la hora de envío y la zona horaria.
- 6 Elige las secciones de contenido que incluye el informe y guarda.

Los resúmenes se envían por el canal de correo; configúralo antes en [Configurar los canales de notificación](#).

Gestionar los resúmenes

La lista muestra la frecuencia, los canales, el próximo envío, el último envío y el estado de cada resumen. Busca en ella, o filtra por estado, frecuencia, canal o fecha de envío.

Para	Haz esto
Detener un resumen durante un tiempo	Ponlo en pausa. Su estado indica Paused hasta que lo vuelvas a activar
Cambiarlo	Ábrelo y edita sus datos
Leer lo que envió	Ábrelo y después su pestaña Generated Reports

Leer un informe

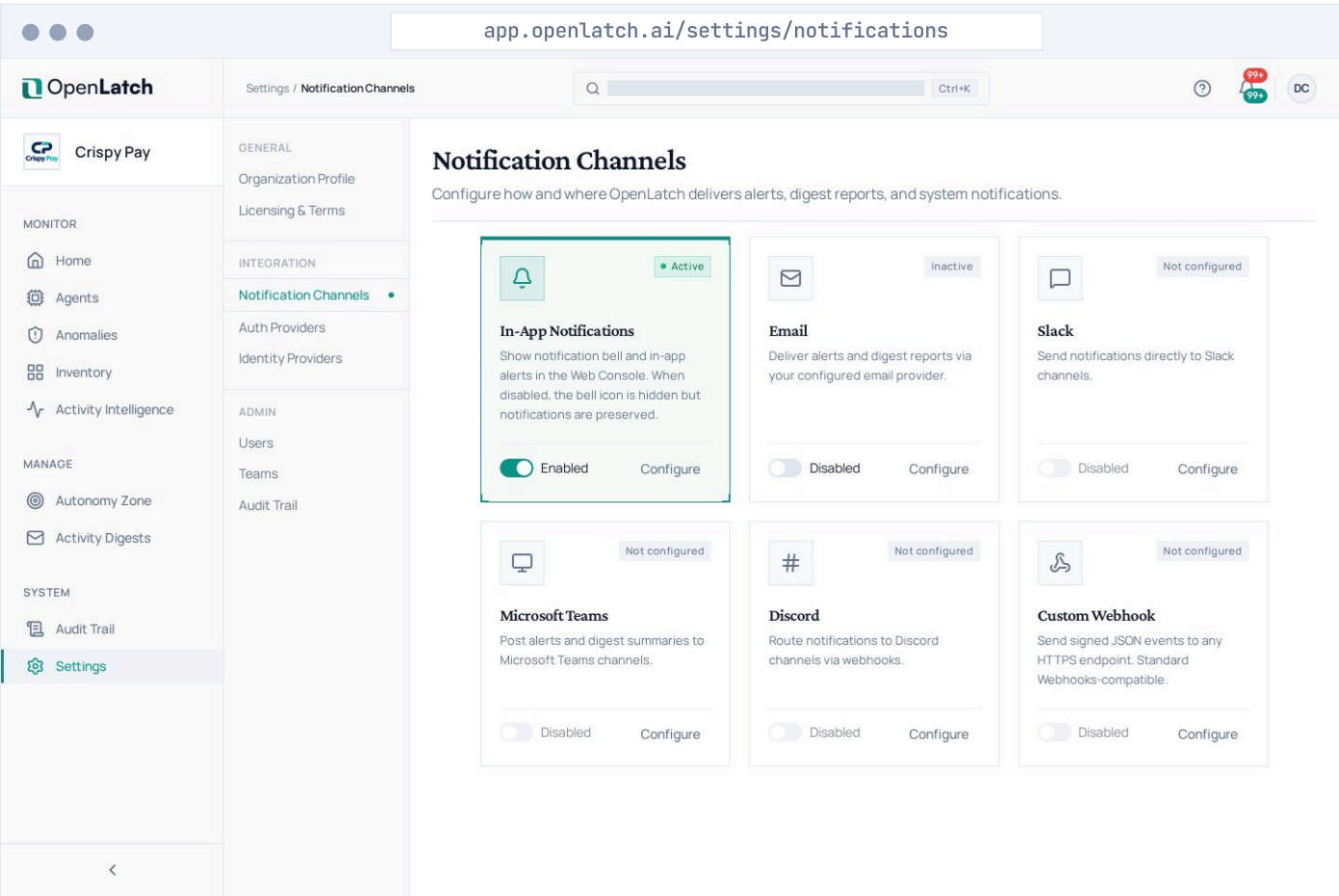
Un informe generado se abre en su propia página, con el formato con el que se envía: un resumen en lenguaje sencillo del periodo, las cifras que lo respaldan y una sección que explica cómo se obtuvo cada cifra y qué significa cada término.



Un informe de resumen: el resumen del periodo, las cifras y cómo se obtuvo cada una.

Configurar los canales de notificación

Usa **Settings > Notification Channels** para elegir cómo y dónde entrega OpenLatch las notificaciones y los informes de resumen. Necesitas el rol admin o superior.



Notification Channels: cada canal al que OpenLatch puede entregar, y si está configurado.

Canales

Canal	Qué hace
Notificaciones en la aplicación	Muestra la campana de notificaciones en la consola. Desactivarlo oculta la campana; las notificaciones se conservan
Correo	Entrega las notificaciones y los informes de resumen mediante el proveedor de correo configurado en tu sitio de OpenLatch
Slack	Envía las notificaciones a canales de Slack
Microsoft Teams	Publica las notificaciones y los resúmenes en canales de Microsoft Teams
Discord	Lleva las notificaciones a canales de Discord mediante webhooks
Webhook personalizado	Envía eventos JSON firmados a cualquier punto de acceso HTTPS, compatible con la especificación Standard Webhooks

Los canales de Slack, Microsoft Teams, Discord y webhook personalizado se pueden añadir varias veces cada uno, por ejemplo uno por equipo.

Añadir un canal

- 1 En la tarjeta del canal, elige **Configure**.
- 2 Sigue la guía de configuración del panel e introduce el destino: una dirección de webhook de Slack, Microsoft Teams o Discord, o tu propio punto de acceso HTTPS.
- 3 Elige qué notificaciones recibe el canal y guarda.
- 4 Envía una prueba y comprueba que llega.

Verificar las firmas de los webhooks

Un webhook personalizado firma cada evento con un secreto de firma, para que tu punto de acceso pueda comprobar que un evento procede de OpenLatch. El secreto se muestra una sola vez, cuando lo revelas para copiarlo en la configuración de tu punto de acceso. **Rotate signing secret** emite uno nuevo; actualiza tu punto de acceso al mismo tiempo.

Para enviar un informe periódico en lugar de notificaciones individuales, consulta [Programar resúmenes de actividad](#).

Revisar el registro de auditoría

Usa la página **Audit Trail** para revisar cada cambio y cada acción de tu organización: quién hizo qué, sobre qué y cuándo. El registro es de solo anexo y está encadenado por hash, así que una entrada modificada o eliminada rompe la cadena. Sus ajustes, en **Settings > Audit Trail**, requieren el rol admin o superior.

app.openlatch.ai/audit-trail

OpenLatch

Manage / Audit Trail

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

Audit Trail

Review all changes and actions across your organization.

Q

Search by actor, target, or changes..

+

Source

+

Action

+

Target Type

+

Status

📅

Date Range

☐	TIMESTAMP ↓	ACTOR ↑	SOURCE ↑	ACTION ↑	TARGET ↑	STATUS ↑	CHANGES
☐		User Documentation user-doc@openlatch.ai	Auth	Login	Session user-doc@openlatch.ai	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Luc Delsalle luc.delsalle@openlatch.ai	API	Directory Providi	Directory Provider Corporate Dir...	Success	
☐		Luc Delsalle luc.delsalle@openlatch.ai	API	Directory Providi	Directory Provider Corporate Dir...	Success	
☐		User Documentation user-doc@openlatch.ai	Auth	Login	Session user-doc@openlatch.ai	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Docs Capture Owner docs-capture@crispypayex...	Auth	Login	Session docs-capture@crispyp...	Success	
☐		Luc Delsalle					

Showing 1 to 25 of 815 events

<

1

2

...

33

>

La página Audit Trail: cada cambio y cada acción, con su autor, su objetivo y su estado.

Encontrar un evento

1

Abre **Audit Trail**.

2

Busca por autor, objetivo o cambio, o filtra por origen, acción, tipo de objetivo, estado e intervalo de fechas.

- 3 Abre un evento. Su detalle muestra el autor, el contexto, el estado, cada campo cambiado con su valor anterior y el nuevo, y el lugar del evento en la cadena de integridad.

Cada cambio de política aparece aquí con su autor y el estado antes y después, igual que cada respuesta dada a un Ask.

Exportarlo

Elige **Export** para descargar los eventos que coinciden con tu búsqueda y tus filtros, en CSV o JSON. La propia exportación queda registrada como evento de auditoría.

Definir la conservación y comprobar la integridad

Abre **Settings** > **Audit Trail**.

The screenshot shows the OpenLatch web interface at the URL `app.openlatch.ai/settings/audit-trail`. The left sidebar contains the OpenLatch logo, the organization name 'Crispy Pay', and a navigation menu with sections: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The main content area is titled 'Audit Trail' and includes a subtitle: 'Configure retention policies, data management, and compliance settings for the audit trail.' The settings are organized into three sections: 1. Retention Policy: Includes a 'Retention Period' selector with options for 90 days, 180 days, 365 days (recommended), 730 days, and Custom. A note states: 'SOC 2 recommends minimum 365 days. GDPR requires documented justification for the chosen period.' Below this is the 'Auto-Purge' toggle, which is enabled, with a note: 'When enabled, audit events older than the retention period are automatically removed every 24 hours.' 2. Export Preferences: A section with a download icon and the title 'Export Preferences'. It contains a note: 'Choose the default format for audit trail exports.' and two radio button options: 'CSV — Spreadsheet-compatible, human-readable' (selected) and 'JSON — Machine-readable, preserves nested data'. 3. Hash Chain Integrity: A section with a shield icon and the title 'Hash Chain Integrity'. It includes a 'Verify Integrity' button and a note: 'Verify the cryptographic hash chain that ensures audit events have not been tampered with.' At the bottom right of the settings area are 'Cancel' and 'Save Configuration' buttons.

Los ajustes Audit Trail: conservación, purga automática, formato de exportación y comprobación de integridad.

Ajuste	Qué hace
Periodo de conservación	Cuánto tiempo se conservan los eventos: 90, 180, 365 o 730 días, o un periodo personalizado. La consola recomienda 365 días
Purga automática	Si está activada, los eventos más antiguos que el periodo de conservación se eliminan cada 24 horas
Formato de exportación	El formato por defecto de una exportación: CSV, legible en una hoja de cálculo, o JSON, que conserva los datos anidados
Verify integrity	Comprueba la cadena de hash de cada evento. Una purga rompe la cadena donde eliminó eventos, y la comprobación informa de esa ruptura

La misma página tiene dos acciones irreversibles: anonimizar el registro, que sustituye el correo, el nombre y el identificador de usuario del actor de cada evento por valores anonimizados conservando los registros y la cadena (las direcciones IP y los demás detalles de cada evento quedan tal como se registraron), y eliminar todos los eventos de auditoría, que rompe la cadena. Cada una te pide escribir una palabra de confirmación.

El registro de auditoría es el único registro cuya conservación defines tú. [Tratamiento de datos y seguridad](#) enumera la conservación fija de todos los demás.

Gestionar usuarios y equipos

Usa **Settings > Users** para decidir quién puede usar la consola y qué puede hacer cada persona, y **Settings > Teams** para agrupar a las personas. Necesitas el rol admin o superior.

Roles

Cuatro roles siguen un orden estricto de privilegios, y cada uno incluye todo lo que está por debajo:

Rol	Qué permite
owner	Propiedad completa: todos los permisos, más la gestión de equipos y de la organización
admin	Acceso completo: usuarios, equipos, ajustes y todo el contenido
editor	Gestiona recursos y configuración, como las políticas de intención, las zonas y la propiedad de los agentes
viewer	Acceso de solo lectura a recursos y configuración

Cada persona invitada empieza como viewer. Concede los roles superiores de forma deliberada, a las pocas personas que los necesitan. La **Role Permissions Reference** de la página enumera cada permiso por rol.

Invitar a alguien

- 1 Abre **Settings > Users** y elige **Invite User**.
- 2 Introduce su dirección de correo y el rol con el que empieza, y envía la invitación.
- 3 La persona recibe un correo con un enlace de registro ligado a esa dirección. También puedes copiar el enlace de invitación y compartirlo tú.

Una invitación puede caducar o revocarse, y cada una sirve una sola vez. Cualquiera con una dirección en uno de los dominios de correo permitidos de tu organización también puede unirse mediante el inicio de sesión único sin invitación; consulta [Configurar el inicio de sesión único](#).

The screenshot displays the 'Users' management page in the OpenLATCH application. The page is titled 'app.openlatch.ai/settings/users'. On the left, there is a sidebar with the 'OpenLATCH' logo and a 'Crispy Pay' badge. The sidebar contains sections for 'MONITOR' (Home, Agents, Anomalies, Inventory, Activity Intelligence), 'MANAGE' (Autonomy Zone, Activity Digests), and 'SYSTEM' (Audit Trail, Settings). The 'Settings' option is currently selected. The main content area is titled 'Users' and includes a subtitle 'Manage team members and invitations.' and an 'Invite User' button. Below this is a search bar and several filter buttons: Status, Team, Email, Created, and Last Login. A table lists the users with columns for Name, Email, Teams, Role, Status, Last Login, and Created. The users listed are: Docs Capture Owner (Owner, Active), Rohan Kapoor (Policy Editor, Active), Camille Lefèvre (Policy Editor, Active), Marcus Washington (Policy Editor, Active), Ingrid Solberg (Admin, Active), User Documentation (Admin, Active), and Luc Delsalle (Owner, Active). At the bottom, it says 'Showing 1 to 7 of 7 users'.

Settings > Users: cada miembro con su rol y su estado.

Cambiar, desactivar o retirar a alguien

Abre un usuario para cambiar su rol o sus datos, o para restablecer su contraseña. Selecciona varios usuarios para activarlos, desactivarlos o eliminarlos a la vez. Un usuario desactivado conserva su historial y no puede iniciar sesión. Cada cambio de rol o de pertenencia queda en el registro de auditoría.

Organizar a las personas en equipos

Los equipos agrupan usuarios, por ejemplo por departamento o por los agentes de los que se ocupan.

- 1 Abre **Settings > Teams** y elige **Create Team**.
- 2 Ponle nombre, describe de qué es responsable y añade sus miembros.

app.openlatch.ai/settings/teams

OpenLatch

Settings / Teams

Q

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

GENERAL

Organization Profile

Licensing & Terms

INTEGRATION

Notification Channels

Auth Providers

Identity Providers

ADMIN

Users

Teams

Audit Trail

Teams

+ Create Team

Organize users into teams with shared permissions.

Q

Search by name or description...

+

Role

+

Members

📅

Created

☐	NAME ↑	DESCRIPTION ↕	ROLE ↕	MEMBERS ↕	CREATED ↕	
☐	Administrators 🔒 System	Organization administrators	Admin	4		⋮
☐	Policy Editors	—	Editor	3		⋮
☐	Users 🔒 System	Default user team	Viewer	0		⋮

Showing 1 to 3 of 3 teams

🔍

ROLE PERMISSIONS REFERENCE

▼

Settings > Teams: cada equipo, su descripción y sus miembros.

Eliminar un equipo elimina sus pertenencias; los usuarios siguen en la organización.

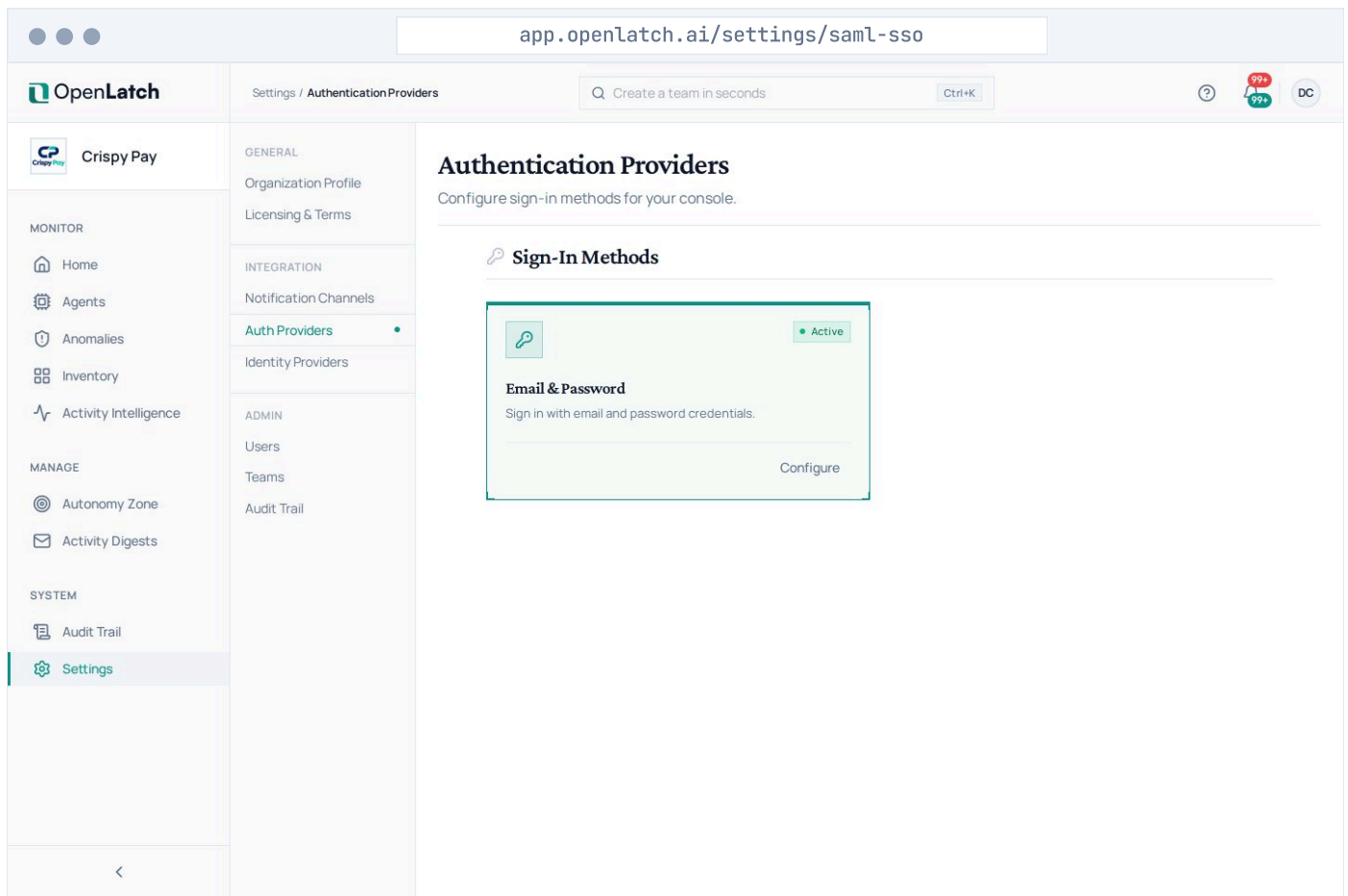
Configurar el inicio de sesión único

Usa **Settings > SAML SSO** para conectar el proveedor de identidad de tu empresa, y **Settings > Auth Providers** para revisar los métodos de inicio de sesión de tu organización y exigir la autenticación en dos pasos. Necesitas el rol admin o superior.

Conectar el inicio de sesión único SAML

La consola admite SAML 2.0, configurado por organización. Microsoft Entra ID, Okta, PingOne y Active Directory Federation Services se conectan así.

- 1 Abre **Settings > SAML SSO**. Copia los datos del proveedor de servicios en tu proveedor de identidad cuando crees allí la aplicación SAML: la URL del Assertion Consumer Service (ACS), el identificador de entidad SP y la URL de metadatos SP.
- 2 Selecciona tu proveedor de identidad para obtener instrucciones escritas para él e introduce sus datos: su identificador de entidad, su URL de inicio de sesión y su certificado de firma o su URL de metadatos. La aserción de tu proveedor de identidad debe incluir la dirección de correo y el nombre visible del usuario.
- 3 Añade los dominios de correo permitidos. Las personas con una dirección en uno de ellos pueden iniciar sesión con SAML. Un dominio solo puede pertenecer a una organización; la consola te avisa si otra organización ya lo ha reclamado.
- 4 Guarda y ejecuta la prueba de conexión. Comprueba, con la configuración guardada, que la URL de metadatos o la URL de inicio de sesión de tu proveedor de identidad responde; no realiza un inicio de sesión SAML. Después inicia sesión tú mismo por SAML, en una ventana privada del navegador, antes de pedírselo a nadie más.

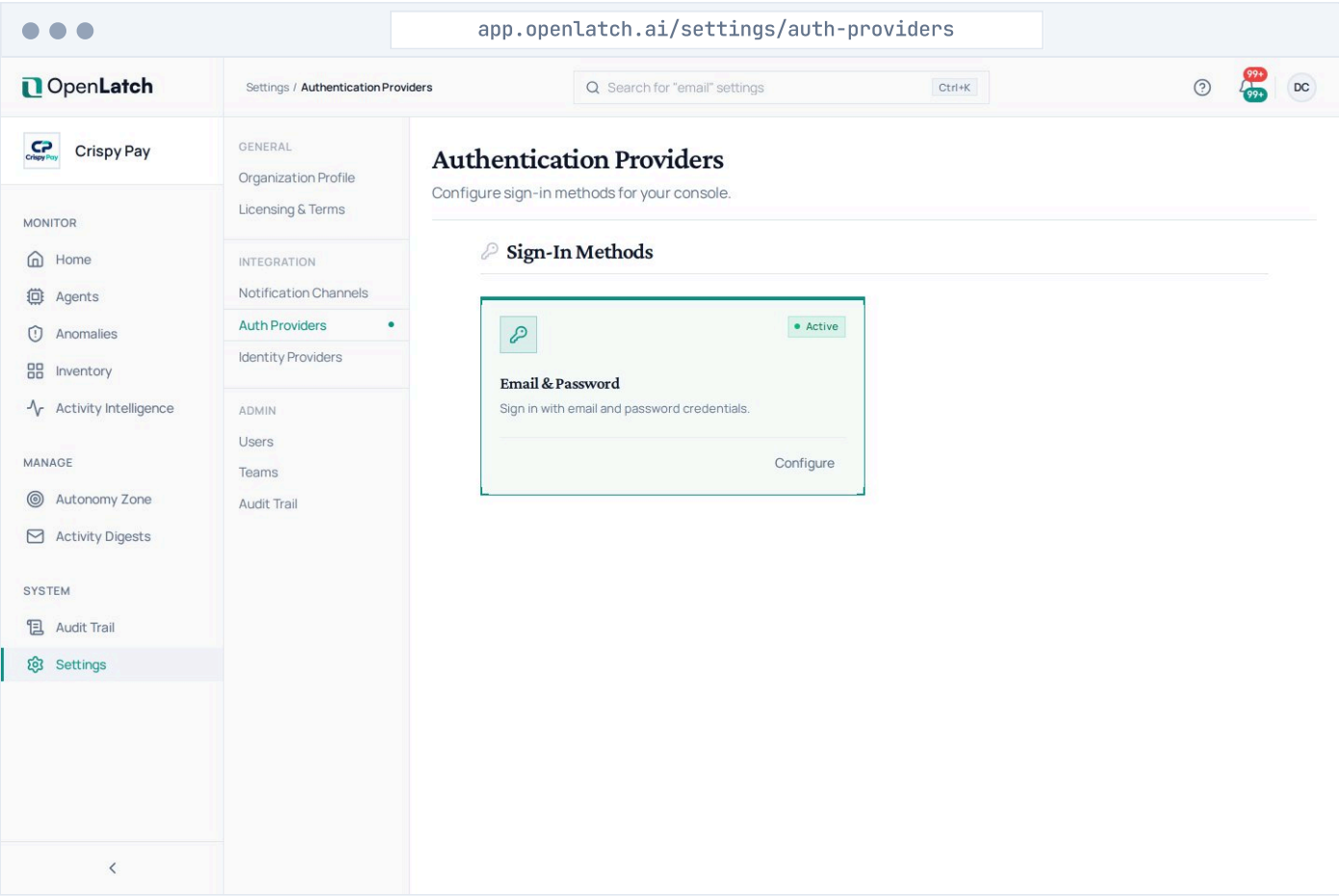


Settings > SAML SSO: los datos del proveedor de servicios que hay que copiar y los de tu proveedor de identidad.

Una conexión SAML propia de la organización es el único método de inicio de sesión que admite a un miembro nuevo sin invitación, y solo para los dominios permitidos.

Revisar los métodos de inicio de sesión y exigir la autenticación en dos pasos

Settings > Auth Providers muestra los métodos de inicio de sesión de tu organización como tarjetas, cada una con su estado: **Email & Password**, y **SAML SSO** cuando SAML está disponible en tu sitio de OpenLatch. **Configure** en una tarjeta abre sus ajustes.



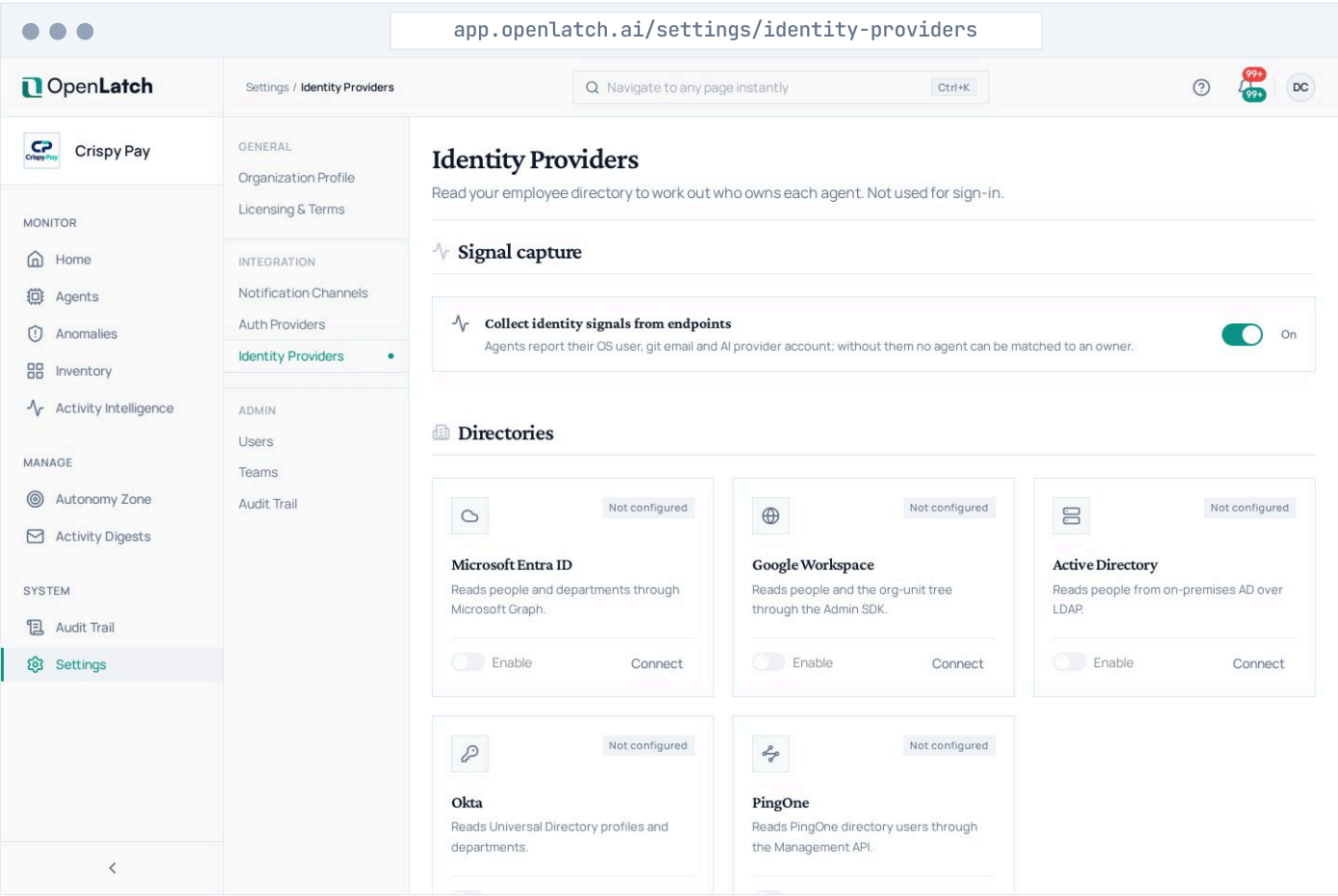
Settings > Auth Providers: los métodos de inicio de sesión que permite tu organización.

Tarjeta	Configure abre
Email & Password	La política de autenticación en dos pasos. Exígela para pedir a cada usuario con contraseña que no la tenga que la configure. Tras el periodo de gracia que elijas, se le envía a sus ajustes de seguridad en cada inicio de sesión hasta que lo haga. Los usuarios que solo inician sesión mediante un proveedor externo quedan exentos. La tarjeta muestra MFA Required mientras la política está activa
SAML SSO	La conexión SAML, configurada como se indica arriba

Cada persona configura su propia autenticación en dos pasos desde su cuenta; consulta [Gestionar tu cuenta](#).

Conectar un directorio para buscar propietarios

Usa **Settings > Identity Providers** para leer tu directorio de empleados, de modo que OpenLatch determine quién es el propietario de cada agente y dónde está esa persona en tu organización. La conexión es de solo lectura y no se usa para iniciar sesión; el inicio de sesión está en [Configurar el inicio de sesión único](#). Necesitas el rol admin o superior para cambiarla.



Settings > Identity Providers: el interruptor de captura de señales y los directorios conectados.

La propiedad se determina desde dos lados, y los dos deben estar presentes:

Lado	Qué aporta	Dónde
Captura de señales	Las máquinas envían las señales que identifican quién trabaja: el usuario del sistema operativo, el correo de git y la cuenta del proveedor de IA	El interruptor Collect identity signals from endpoints , activado por defecto
Directorios	El directorio convierte esas señales en una persona con nombre, con su cargo y su lugar en la organización	Las tarjetas de directorio

Con la captura de señales desactivada, ningún agente puede asociarse a un propietario, por bien conectado que esté el directorio.

Conectar un directorio

- 1 Abre **Settings > Identity Providers**.
- 2 Elige la tarjeta de tu directorio: Microsoft Entra ID, Google Workspace, Active Directory, Okta o PingOne. Cada uno es una fuente de solo lectura: ninguno crea usuarios en OpenLatch y ninguno copia el directorio en él.
- 3 Sigue los pasos del panel para conceder a OpenLatch acceso de lectura en tu directorio y guarda. Si al directorio le falta un permiso que OpenLatch necesita, la tarjeta indica cuál.
- 4 Comprueba el resultado en un agente que conozcas: su pestaña **Identity** debe nombrar a su propietario. Consulta [Encontrar y leer un agente](#).

Una organización sin directorio conectado es un estado admitido. Un agente cuyo propietario no puede determinarse se registra como sin atribuir en lugar de suponerse, y un editor puede definir su propietario a mano desde su pestaña **Identity**.

Lo que también aporta el directorio

Las unidades del directorio forman el árbol de tu organización en la página **Autonomy Zone**, y cada agente está por defecto en la unidad de su propietario. Una unidad que procede del directorio conserva el nombre y la posición que le da el directorio, y se actualiza en cada sincronización; la Autonomy Zone que le pongas sigue siendo tuya. Consulta [Gestionar las Autonomy Zones](#).

Desconectar un directorio

Desconectar un directorio elimina su conexión, sus credenciales guardadas y las personas que había determinado. Los agentes asociados mediante él pierden su propietario hasta que otro directorio los determine.

La dirección `/settings/directory-sync` es un nombre anterior de esta página y la abre.

Editar el perfil de la organización

Usa **Settings > Organization Profile** para gestionar cómo aparece tu organización y los valores por defecto que se aplican a toda ella. Abrir **Settings** te lleva a esta página. Necesitas el rol admin o superior para cambiarla.

app.openlatch.ai/settings/organization-profile

OpenLatch

Settings / Organization Profile

Go to Organization Profile

Ctrl+K

?

99+

99+

DC

Crispy Pay

MONITOR

Home

Agents

Anomalies

Inventory

Activity Intelligence

MANAGE

Autonomy Zone

Activity Digests

SYSTEM

Audit Trail

Settings

GENERAL

Organization Profile

Licensing & Terms

INTEGRATION

Notification Channels

Auth Providers

Identity Providers

ADMIN

Users

Teams

Audit Trail

Organization Profile

Manage your organization's public identity and primary domain.

Identity

Organization Logo

Recommended size 400x400px

Change Logo

Remove

Organization Name

Crispy Pay

The legal name of the entity operating this property.

System Metadata

ORGANIZATION ID

208bbd63-67fd-49c5-9bdf-9c6d55dd7f72

ORGANIZATION SLUG

crispy-pay

CREATED AT

SITE

us-east-1 (United States)

Your organization's data is stored in this region. OpenLatch's residency statement is available on request, in the trust pack and the DPA.

Agent Budgets

Reset to Defaults

Save Changes

Settings > Organization Profile: la identidad de la organización, sus metadatos del sistema y su configuración.

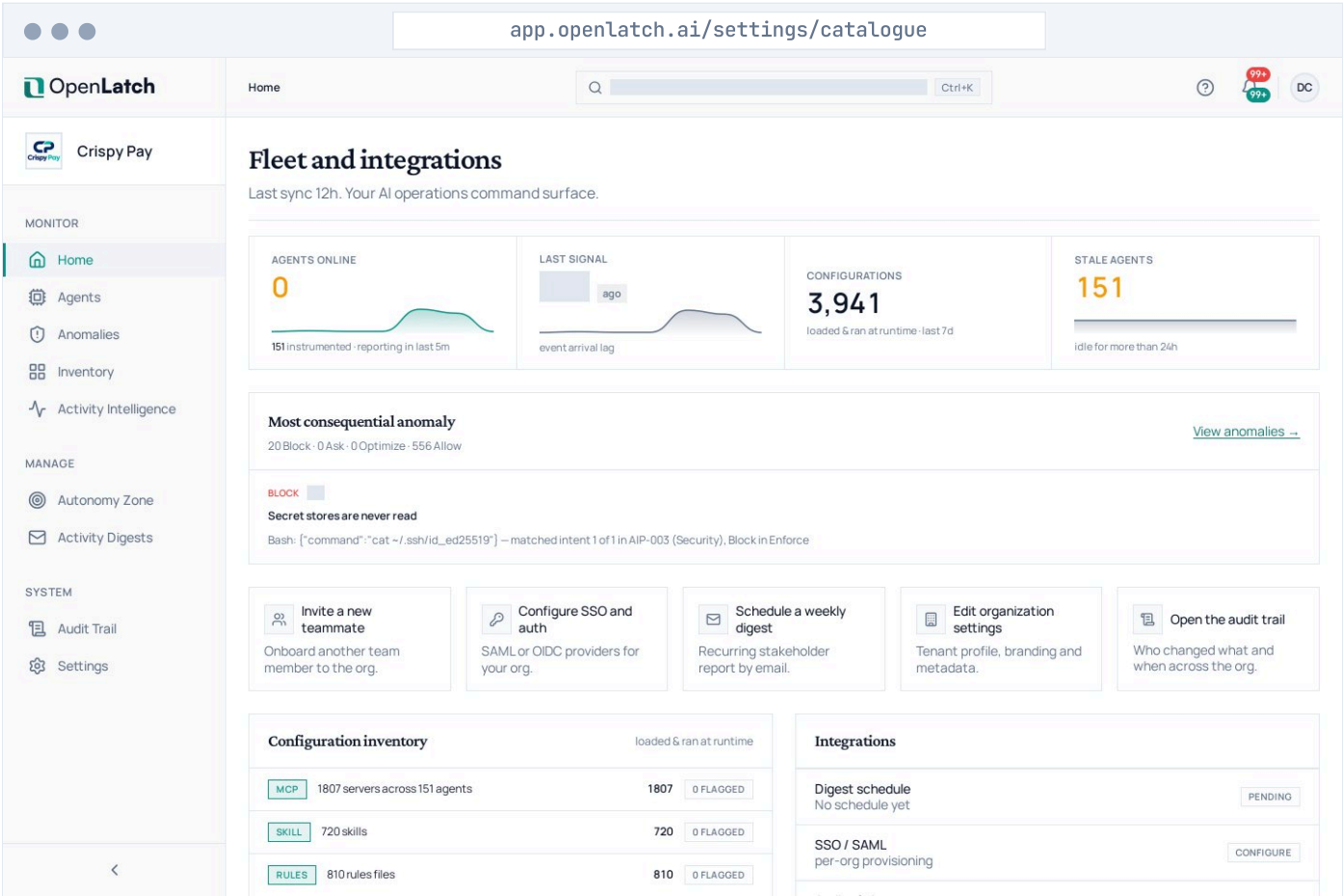
Sección	Qué puedes hacer
Identidad	Cambiar el nombre de la organización, la razón social de la entidad que representa, y subir o quitar su logotipo, idealmente de 400×400 píxeles. Ese nombre es el que lleva tu registro de auditoría
Metadatos del sistema	Leer el identificador de la organización, su slug, cuándo se creó y su sitio: la región donde se almacenan los datos de tu organización. Indica el identificador cuando contactes con soporte
Presupuestos de los agentes	Definir un objetivo de gasto mensual por defecto para cada agente sin presupuesto propio. Ese valor se aplica a cada uno de esos agentes por separado: un valor por defecto de 500 \$ da a cada uno 500 \$ al mes, no 500 \$ compartidos. Borrarlo nunca sobrescribe un presupuesto definido en un agente
Configuración	Volver a ejecutar el asistente de configuración para revisar la conexión y el perfil de la organización

Guarda los cambios con **Save Changes**. El presupuesto propio de un agente se define desde su pestaña **Economics**; consulta [Seguir el gasto de un agente](#).

Las demás páginas de ajustes tienen cada una su guía: [usuarios y equipos](#), [inicio de sesión único](#), [proveedores de identidad](#), [canales de notificación](#), el [registro de auditoría](#), el [catálogo](#) y las [licencias](#).

Ajustar el catálogo

Usa **Settings > Catalogue** para ver y ajustar cómo reconoce OpenLatch lo que hace una acción de agente y lo que toca. Las políticas de intención se escriben en términos de lo que hacen las acciones, como leer una credencial o subir código; el catálogo es lo que asocia una herramienta, un programa o un destino concretos a esos términos. Todos los roles pueden leerlo; solo un admin puede añadir o cambiar entradas.



Settings > Catalogue: las herramientas, programas y destinos que reconoce OpenLatch, con el efecto y el origen de cada entrada.

Leer el catálogo

El catálogo tiene tres pestañas:

Pestaña	Qué reconocen sus entradas
Tools	Las herramientas que llama un agente
Programs	Los programas que ejecuta un agente en un shell
Targets	Lo que toca una acción, como un archivo, un host o un repositorio

Cada entrada muestra su clave, su efecto, su origen y cuándo se añadió. Las entradas tienen varios orígenes: la base de OpenLatch, el aprendizaje a partir de las acciones de tus agentes, la redacción de políticas o un admin. **Needs review** filtra las entradas que esperan una decisión, y **Show OpenLatch seed** añade a la lista las entradas de base, de solo lectura.

Las entradas aprendidas se aplican a una política de intención según el modo de cobertura de esa política; consulta [Redactar una Agent Intent Policy](#). Un catálogo vacío se va llenando a medida que OpenLatch aprende las herramientas, programas y destinos que usan tus agentes.

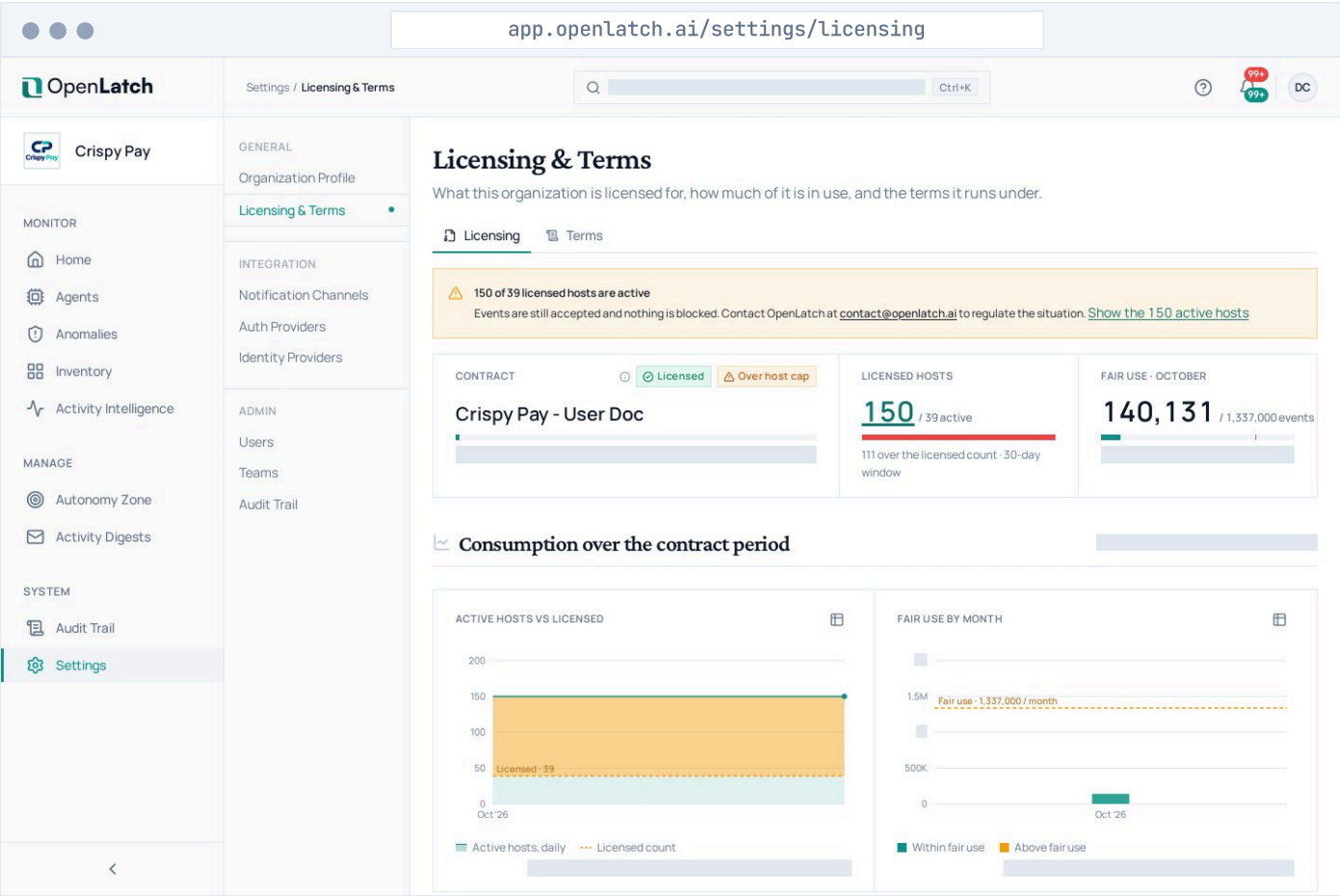
Ajustar una entrada

Para	Haz esto
Añadir lo que OpenLatch aún no reconoce	Elige Add entry e indica la clave y su efecto
Cambiar el significado de una entrada aprendida	Elige Override en la entrada
Hacer más estricta una entrada de base para tu organización	Elige Narrow en ella. La base en sí es de solo lectura; tu entrada acota su clave

Consultar licencias y uso

Usa **Settings > Licensing** para ver para qué tiene licencia tu organización, cuánto está en uso y las condiciones con las que funciona. Retirar un host requiere el rol admin o superior.

Tu licencia cuenta hosts: las máquinas que ejecutan el cliente. Cada licencia incluye también una asignación mensual de uso razonable, en eventos.



Settings > Licensing: el contrato, los hosts con licencia en uso y el uso razonable del mes.

Leer los contadores

La pestaña **Licensing** se abre con tres contadores:

Contador	Qué muestra
Contrato	El periodo de tus condiciones y cuánto ha transcurrido
Hosts con licencia	Cuántos hosts están activos, es decir, vistos en los últimos 30 días, frente al número con licencia
Uso razonable	Los eventos usados en este mes natural, frente a la asignación mensual. Se reinicia al principio de cada mes, en UTC

Solo aparece un aviso cuando algo requiere tu atención: las condiciones aún no han empezado, el contrato termina pronto, hay más hosts activos que licencias, o el uso razonable del mes llega al 100 %. Superar los hosts con licencia o la asignación de uso razonable no bloquea nada; los eventos se siguen aceptando, y el aviso te pide que contactes con OpenLatch en contact@openlatch.ai. Las fechas del contrato son distintas. Antes de que empiecen tus condiciones, la consola está disponible pero se rechazan los eventos del cliente. Cuando el contrato ha terminado, se rechazan los eventos del cliente y la consola queda restringida hasta que se renuevan las condiciones; contacta con OpenLatch en contact@openlatch.ai antes de la fecha de fin para renovar.

Debajo de los contadores, los gráficos muestran los hosts activos frente al número con licencia y el uso razonable mes a mes durante el periodo del contrato, y la pestaña **Terms** contiene las propias condiciones.

Encontrar un host

La tabla de hosts con licencia enumera cada host con su estado, su sistema operativo, su propietario y su última actividad. Fíltrala por estado, sistema operativo, propietario o última actividad, o busca en ella. Abre un host para ver los agentes que se ejecutan en él.

Retirar un host

Retira una máquina que ya no ejecuta el cliente, para que deje de contar entre tus hosts con licencia.

- 1 Selecciona el host, o varios hosts, en la tabla.
- 2 Elige **Retire host** y confirma.

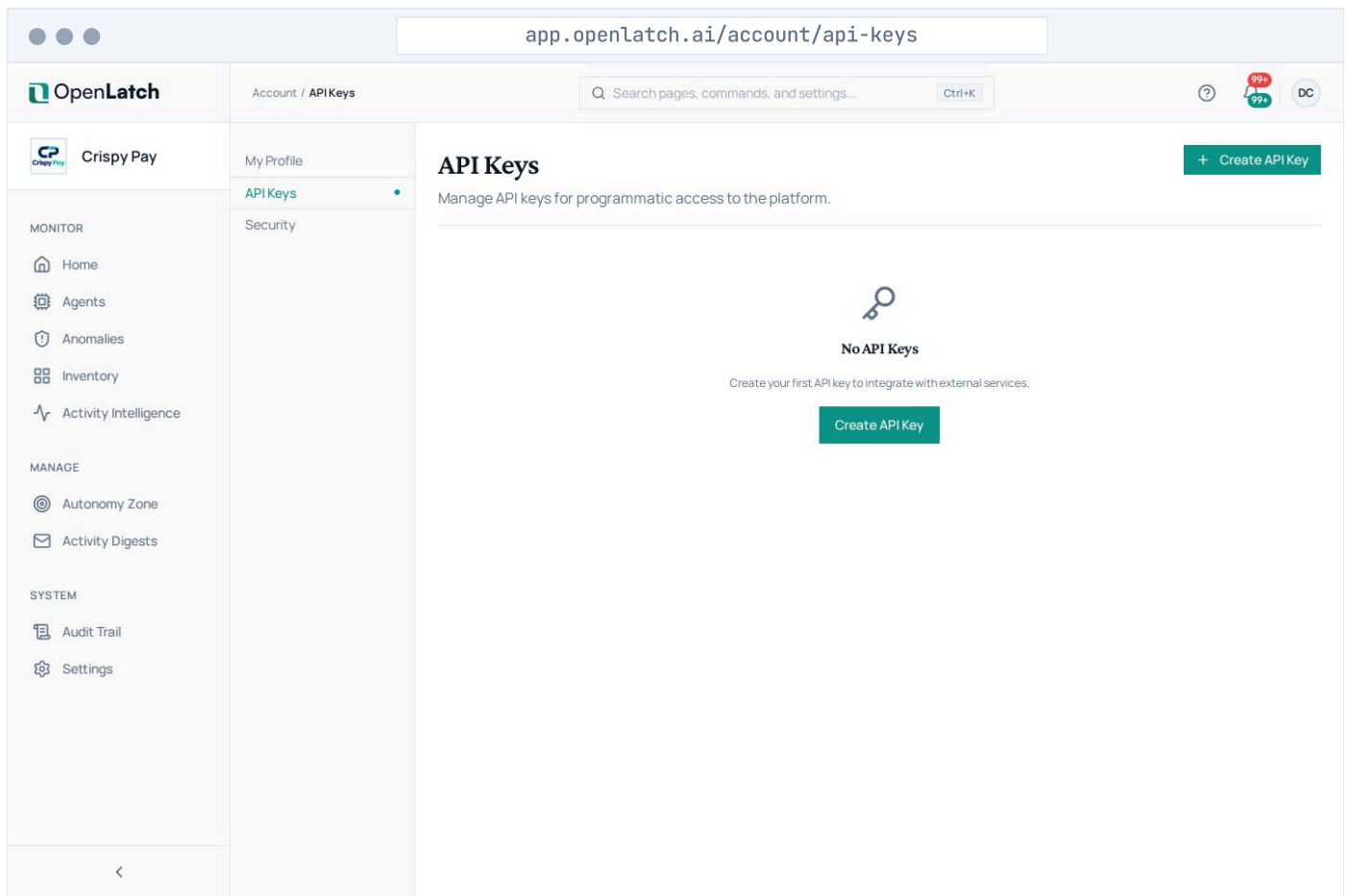
Un host retirado deja de contar al instante. Su historial de eventos se conserva, y la retirada queda en el registro de auditoría con tu nombre. Si el host vuelve a verse, vuelve a contar automáticamente. Para retirar el cliente de una máquina, consulta [Operar](#).

Gestionar las claves de API

Usa **Account > API Keys** para crear las claves que dan a un programa acceso a OpenLatch en tu nombre. Una clave sirve para dos usos:

Uso	Cómo se proporciona la clave
Inscribir una máquina sin iniciar sesión en el navegador, para una instalación gestionada o desatendida	Como OPENLATCH_API_KEY en el entorno de la instalación. Consulta Desplegar el cliente
Llamar a la API REST pública de OpenLatch, bajo /api/public/v1 en app.openlatch.ai	En la cabecera Authorization: Bearer de cada petición

Una clave actúa con tu rol en la organización. Una creada por un editor no puede cambiar un ajuste de identidad, y revocar la clave o rebajar tu rol cierran la misma puerta.



Account > API Keys: cada clave con su nombre, su valor enmascarado, su estado, su fecha de creación y su último uso.

Crear una clave

- 1 Abre la página **API Keys** de tu cuenta y elige **Create API Key**.
- 2 Ponle a la clave el nombre de lo que la usará, por ejemplo la oleada de despliegue o la integración, y elige **Create Key**.
- 3 Copia ahora el valor de la clave. Se muestra una sola vez, completo; después de cerrarlo, la lista solo muestra un valor enmascarado.
- 4 Guárdala en el almacén de secretos de tu organización. Nunca la escribas en un script, un paquete, una línea de comandos ni un documento compartido.

Usa una clave por uso: una por oleada de despliegue, una por integración. Una clave que puedes revocar sola es una clave que puedes revocar sin afectar a nada más.

Revocar una clave

Elige **Revoke** en la clave y confirma. La revocación no se puede deshacer, y todo lo que usa la clave pierde el acceso al instante. Una máquina inscrita con una clave de oleada guarda esa clave y se autentica con ella, así que revocar la clave desconecta la máquina. Da primero una clave nueva a esas máquinas: defínela como `OPENLATCH_API_KEY` y vuelve a ejecutar `openlatch init --yes` en cada una.

La columna **Last used** te indica si una clave se sigue presentando antes de revocarla. Una clave que ha caducado figura como **Expired**.

 Precaución

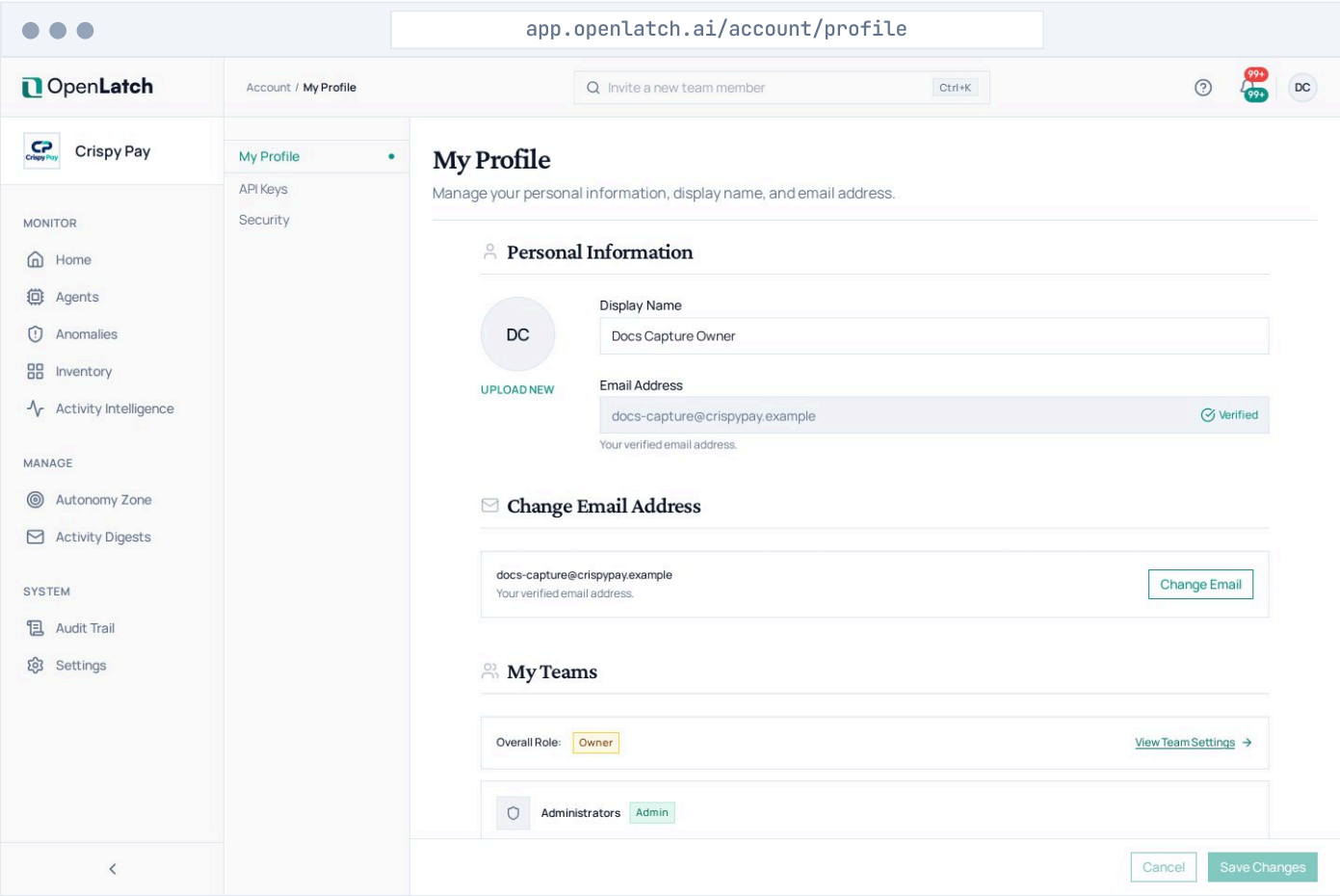
Una clave pasada en una línea de comandos acaba en el historial del shell, en los listados de procesos y en los registros de la gestión de configuración, y el cliente rechaza una credencial pasada así. Proporciona una clave por el entorno, desde tu almacén de secretos.

Gestionar tu cuenta

Los ajustes de tu cuenta son tuyos: todos los roles pueden cambiarlos. Ábrelos desde tu avatar en la consola; **Account** se abre en tu perfil.

Editar tu perfil

Account > Profile contiene tu información personal: tu nombre visible y tu dirección de correo. Cambiar la dirección de correo pide tu contraseña actual.

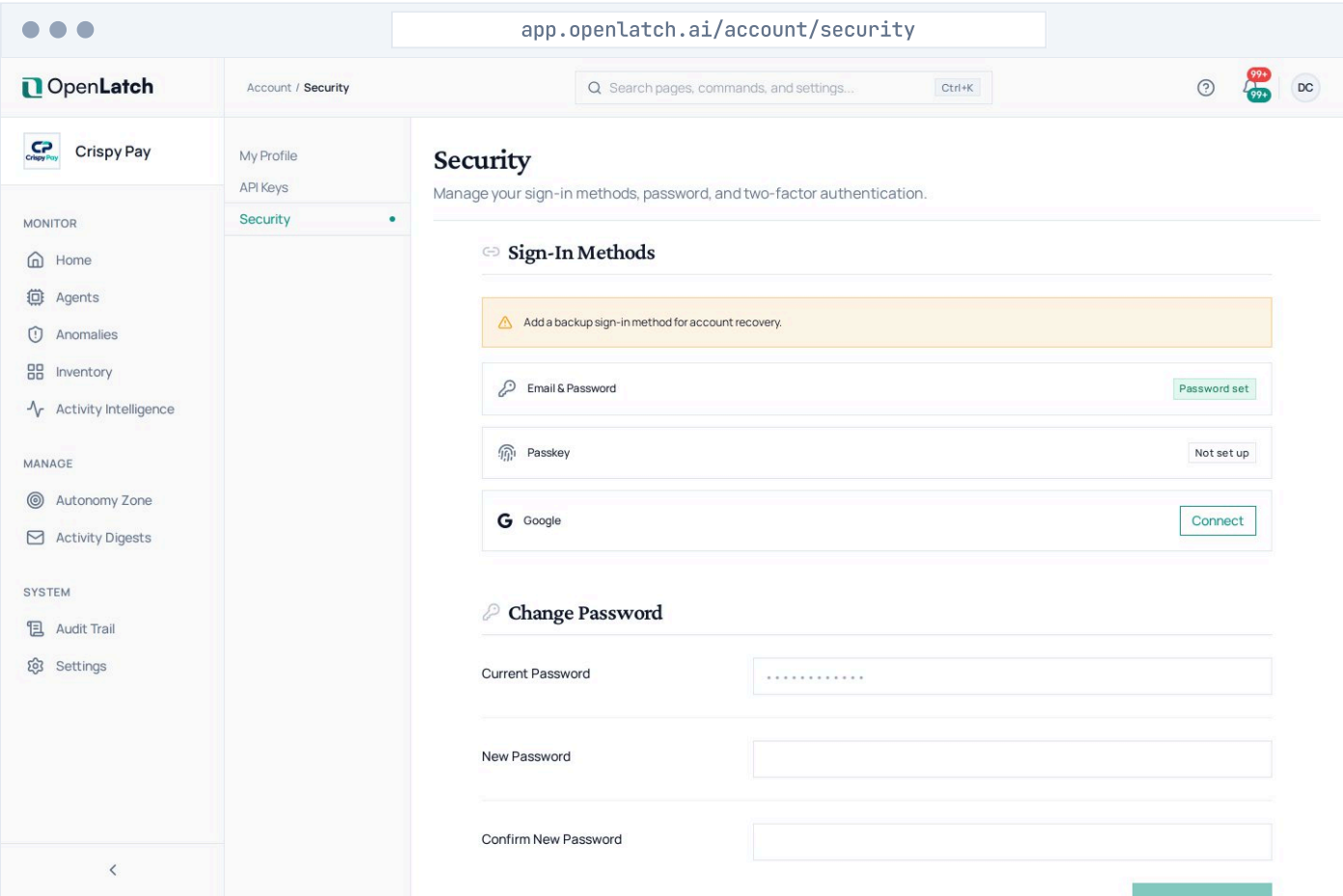


Account > Profile: tu nombre visible y tu dirección de correo.

Tu cuenta también enumera tus sesiones activas en tus dispositivos. Cierra cualquier sesión que no reconozcas.

Proteger tu inicio de sesión

Account > Security gestiona tus métodos de inicio de sesión, tu contraseña y la autenticación en dos pasos.



Account > Security: tus métodos de inicio de sesión, tu contraseña, la autenticación en dos pasos y las passkeys.

Ajuste	Qué hace
Contraseña	Cambiar tu contraseña; la consola pide antes la actual
Autenticación en dos pasos	Añade un código de una aplicación de autenticación a cada inicio de sesión con contraseña
Passkeys	Iniciar sesión con una passkey guardada en tu dispositivo o en tu gestor de contraseñas. Ponle nombre a cada passkey para distinguirlas
Eliminar la cuenta	Elimina tu cuenta. La consola pide tu contraseña para confirmar

Para configurar la autenticación en dos pasos:

- 1 En **Account > Security**, inicia la configuración de la aplicación de autenticación y confirma tu contraseña.
- 2 Escanea el código QR con tu aplicación de autenticación.
- 3 Guarda los códigos de recuperación en un lugar seguro. Cada código sirve una vez, y es como inicias sesión si pierdes el dispositivo.
- 4 Introduce un código de seis dígitos de la aplicación para terminar.

Si tu organización exige la autenticación en dos pasos, la consola te envía a esta página hasta que la configures; consulta [Configurar el inicio de sesión único](#). Los métodos de inicio de sesión que puedes usar los elige tu organización.

Las claves para programas y para inscribir máquinas están en **Account > API Keys**; consulta [Gestionar las claves de API](#).

Desplegar OpenLatch

Esta sección te lleva de cero a un despliegue en funcionamiento: crear la organización, conectar tu proveedor de identidad, desplegar el cliente en cada máquina que ejecuta un agente y preparar la consola para las personas que la usarán. Está escrita para que puedas hacer el trabajo sin una llamada.

Qué cubre la sección

Quieres	Lee
Entender qué es OpenLatch y qué se ejecuta dónde	Introducción · Arquitectura
Preparar: qué tener a mano y el único destino saliente que hay que permitir	Requisitos previos · Requisitos de red
Instalar: la organización, el cliente en cada máquina, la verificación y el despliegue	Configurar la organización · Desplegar el cliente · Verificar y desplegar
Mantenerlo en marcha	Operar
Responder a una revisión de seguridad y a las preguntas que le siguen	Tratamiento de datos y seguridad · Preguntas frecuentes del despliegue
Consultar algo	Términos y tráfico saliente

Cómo usar esta sección

Léela en orden la primera vez. La introducción y la página de arquitectura explican qué es OpenLatch y qué se ejecuta dónde; bastan para responder a una primera revisión antes de instalar nada. Los requisitos previos y los requisitos de red son lo que preparas de antemano: los valores que tener a mano y el único destino saliente que hay que permitir. Las tres páginas siguientes son la instalación en sí, desde el primer inicio de sesión hasta un despliegue verificado, y [Operar](#) cubre lo que viene después.

[Tratamiento de datos y seguridad](#) expone lo que OpenLatch trata, almacena y nunca recopila. Es la página que hay que entregar a una revisión de seguridad. Las [Preguntas frecuentes del despliegue](#) responden a las

preguntas que siguen, y [Términos y tráfico saliente](#) reúne los términos y la tabla de tráfico saliente.

Cada página de esta sección se imprime por sí sola, sin la navegación del sitio, para que puedas adjuntarla a una solicitud de cambio o a una revisión.

Comentarios y avisos de seguridad

Envía correcciones y lagunas a hello@openlatch.ai, indicando la página. Los avisos de seguridad siguen otra vía: envíalos a security@openlatch.ai. OpenLatch confirma la recepción de un aviso de seguridad en un plazo de dos días hábiles.

Introducción

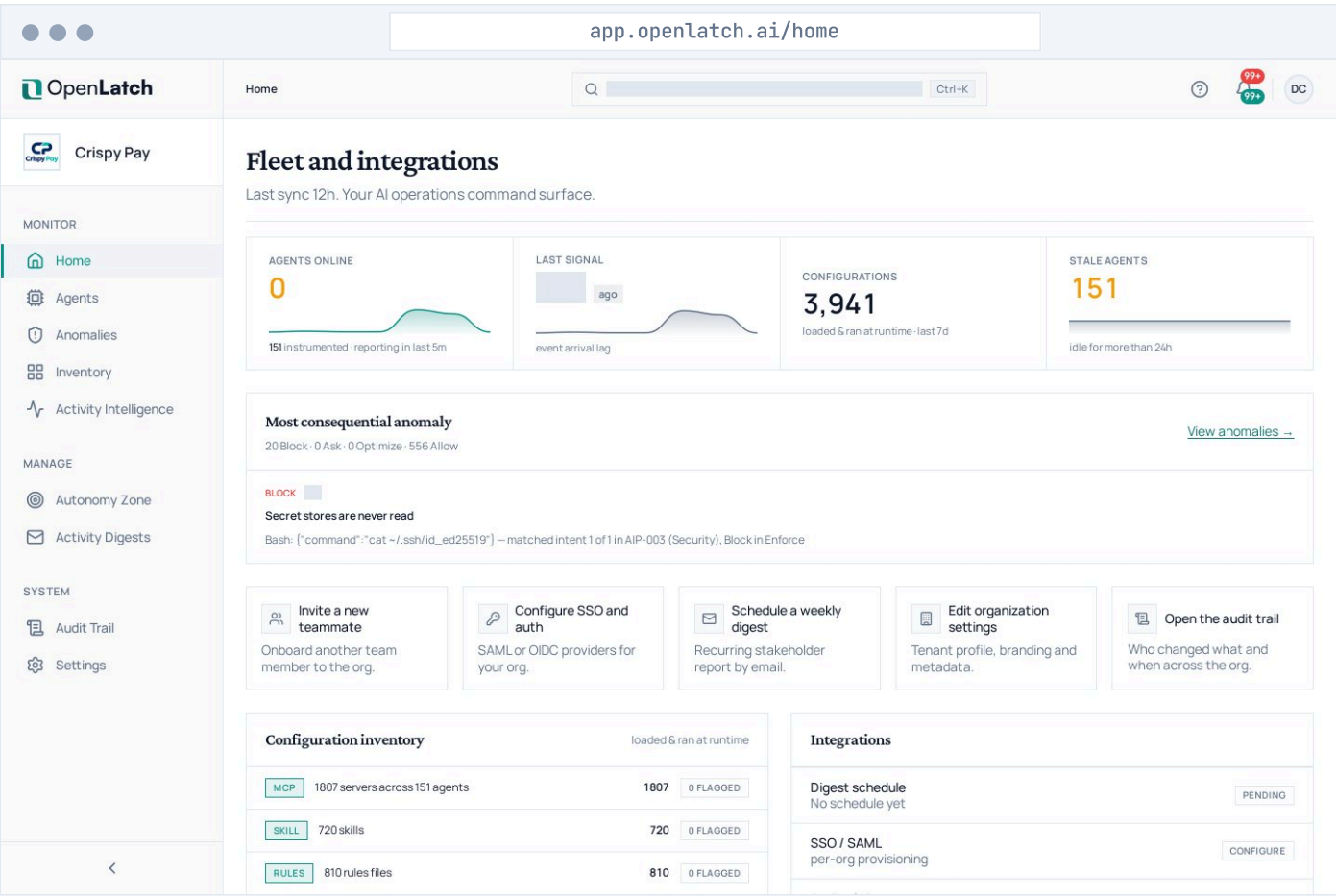
Lee esta página para saber qué vas a desplegar antes de preparar nada.

Qué es OpenLatch

OpenLatch es la capa de control de ejecución para agentes de IA empresariales. Se ejecuta dentro del propio ciclo de vida del agente, evalúa cada acción cubierta frente a las Agent Intent Policies de tu organización y devuelve un veredicto determinista antes de que la acción se ejecute: Allow, Ask, Block u Optimize.

Determinista tiene aquí un significado preciso. Cada acción cubierta llega al punto de decisión de las políticas, y el veredicto que produjo una política explícita es el que se aplica. [Arquitectura](#) muestra el camino que recorre una acción para llegar ahí.

Lo que obtienes es un único lugar que reúne cada agente de la organización, las reglas con las que trabaja cada uno y las decisiones ya tomadas en su nombre.



La consola una vez conectados los agentes y definido por la organización lo que pueden hacer.

Los cuatro veredictos

Veredicto	Qué ocurre
Allow	La acción continúa
Ask	La acción continúa y se marca para revisión asíncrona. Solo aguarda una respuesta cuando una política activa la opción por átomo para acciones irreversibles
Block	La acción se detiene antes de ejecutarse
Optimize	El camino cambia. Aplicado en Enforce, medido en Monitor

[Veredictos](#) explica cada uno en detalle.

Qué se ejecuta dónde

OpenLatch tiene dos partes: una consola en la que trabajas y un cliente que instalas en las máquinas que ejecutan agentes.

Parte	Qué es	Dónde se ejecuta
La consola	Contiene la organización, sus usuarios y roles, el registro de agentes y el registro de auditoría, y es donde se redactan las políticas. Nunca aplica un veredicto por sí misma	Se ofrece como servicio en <code>app.openlatch.ai</code> . El alojamiento y el tratamiento de datos se describen en Tratamiento de datos y seguridad
El cliente	Una única unidad instalada. Evalúa cada acción cubierta en la máquina, devuelve el veredicto antes de que la acción se ejecute y envía el registro de lo ocurrido a la consola. La instalación detecta los agentes presentes en la máquina y conecta cada uno	Cada máquina que ejecuta un agente, una vez por máquina. Se instala y se ejecuta a nivel de usuario, sin derechos de administrador

Lo que el despliegue no le pide a tu red:

Aspecto	Lo que necesita OpenLatch
Puertos entrantes	Ninguno. El cliente solo escucha en la interfaz de bucle local, y nada de la máquina es accesible desde la red
Certificados	Ninguno. OpenLatch no emite certificados ni los exige
Intercepción de tráfico	Ninguna. OpenLatch no termina tráfico ni lo vuelve a cifrar con otro certificado
Cambio de red	Una única entrada saliente que permitir: <code>app.openlatch.ai</code> , en TCP 443
Tu proxy actual	El cliente funciona detrás de él, incluso con inspección TLS cuando la autoridad de certificación de intercepción es de confianza. Consulta Requisitos de red

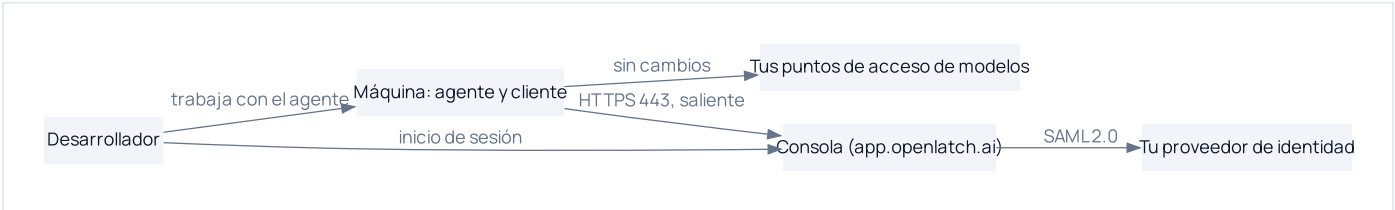


Figura 1. OpenLatch en la organización: el cliente en cada máquina, la consola como servicio y el proveedor de identidad.

Qué vas a configurar

Paso	Qué significa	Dónde
1 · Crear la organización	Iniciar sesión, nombrar la organización, invitar a los administradores y definir sus roles	Configurar la organización
2 · Conectar el proveedor de identidad	Configurar el inicio de sesión único con SAML 2.0 y probar el intercambio antes de invitar a nadie más	Configurar la organización
3 · Conectar los agentes	Desplegar el cliente en cada máquina, por la ruta de desarrollador o con tus herramientas de despliegue gestionado	Desplegar el cliente
4 · Verificar y desplegar	Demostrar que cada máquina está conectada y ampliar el despliegue en tres oleadas	Verificar y desplegar

Qué necesitas para empezar

Requisito	Detalle
Una cuenta en la consola	OpenLatch crea la organización e invita a su propietario. Esa cuenta invita a todos los demás
HTTPS saliente hacia <code>app.openlatch.ai</code> en TCP 443	El único destino que exige OpenLatch, a través de tu proxy. No se abre ningún puerto entrante. Permítelo por nombre, porque no hay ningún rango de direcciones estático

[Requisitos previos](#) convierte esto en un inventario de preparación, con el origen de cada valor.

En resumen

- El veredicto se produce en la máquina, antes de que la acción se ejecute. La consola no está en el camino de decisión.
- Las políticas siguen vigentes en la máquina cuando la consola no es accesible. Los registros se guardan localmente y se envían cuando vuelve a serlo.

- Se necesita un único destino saliente. Sin puertos entrantes, sin certificados que instalar, sin interceptación de tráfico.
- La configuración tiene cuatro pasos: la organización, el proveedor de identidad, el cliente en cada máquina y, por último, la verificación y el despliegue.

Arquitectura

Lee esta página para planificar en función de lo que el cliente instala en una máquina y de lo que cruza la frontera de la máquina.

Qué se ejecuta en la máquina

Aquí, una máquina es el sistema operativo en el que se ejecutan los agentes. En la mayoría de los entornos es un equipo de desarrollador, y lo es igualmente un escritorio virtual, un host de compilación o una imagen de contenedor, dondequiera que se inicie un agente. Si allí se ejecutan agentes, allí se instala el cliente.

OpenLatch instala una sola cosa en esa máquina: el cliente de OpenLatch. Se instala una vez y cubre cada agente de esa máquina. Nada que colocar delante del agente, ningún servicio que levantar a su lado y ningún segundo producto que mantener sincronizado con el primero.

Aspecto	Detalle
Qué se instala	El cliente de OpenLatch, una vez por sistema operativo que ejecuta agentes
Privilegios	Se instala y se ejecuta a nivel de usuario. No se necesitan derechos de administrador, ni en la instalación ni después
Postura de red	Solo escucha en la interfaz de bucle local. No se abre ningún puerto entrante, y nada de la máquina es accesible desde la red
Archivos	El cliente guarda su configuración y sus registros locales en un único directorio: <code>~/ .openlatch</code> en macOS y Linux, <code>%APPDATA%\openlatch</code> en Windows. <code>OPENLATCH_DIR</code> lo cambia de sitio
Arranque	Arranca con la sesión del usuario, bajo el gestor de servicios del sistema operativo
Dónde se toman las decisiones	En la máquina, antes de que la acción se ejecute. La consola no está en ese camino
Pérdida de conectividad	Sigue funcionando. Las políticas siguen vigentes y los registros se guardan hasta que la consola vuelve a ser accesible



Figura 1. Vista de despliegue: el cliente en cada máquina y la consola como servicio.

La consola

La consola se ofrece como servicio en `app.openlatch.ai`. Nada que instalar ni que alojar. Los datos de tu organización se almacenan separados de los de cualquier otra organización.

La consola contiene cuatro elementos con los que trabajas: la organización, sus usuarios y sus roles, el registro de agentes y el registro de auditoría. El registro de agentes enumera cada máquina conectada y los agentes que hay en ella, cada uno con un propietario y un [nivel de control](#): Enforced, Monitored o Known. El registro de auditoría es de solo anexo y exportable, con un periodo de conservación que tú defines.

[Configurar la organización](#) describe el procedimiento para todo esto.

Las políticas se redactan en la consola y desde allí llegan a cada máquina. [Redactar una Agent Intent Policy](#) cubre la redacción.

Cómo se decide una acción

Una acción produce un veredicto, en la máquina, antes de que la acción se ejecute. La consola no está en este camino.

Paso	Qué ocurre	Qué puedes observar
1 · El agente propone la acción	El agente va a ejecutar un comando de shell o a llamar a una herramienta. Primero entrega la acción propuesta a OpenLatch y aguarda	—
2 · OpenLatch decide en la máquina	La decisión se toma frente a las políticas que la máquina ya tiene. No hay llamada de red ni modelo en este camino	Se obtiene un veredicto tanto si la consola es accesible como si no
3 · El veredicto vuelve al agente	El veredicto vuelve en la forma que el agente entiende, con su motivo. Cuando coinciden varias políticas, gana la más restrictiva	Ante un Block, el desarrollador ve la negativa y su motivo en el agente, y la acción no se ejecuta
4 · El resultado se registra	La acción y su veredicto se anotan en la máquina y se envían después a la consola. El envío está fuera del camino de decisión: una consola lenta cambia cuándo se ve una acción, nunca lo que se decidió	La máquina, sus agentes y sus acciones decididas aparecen en el registro de agentes de la consola

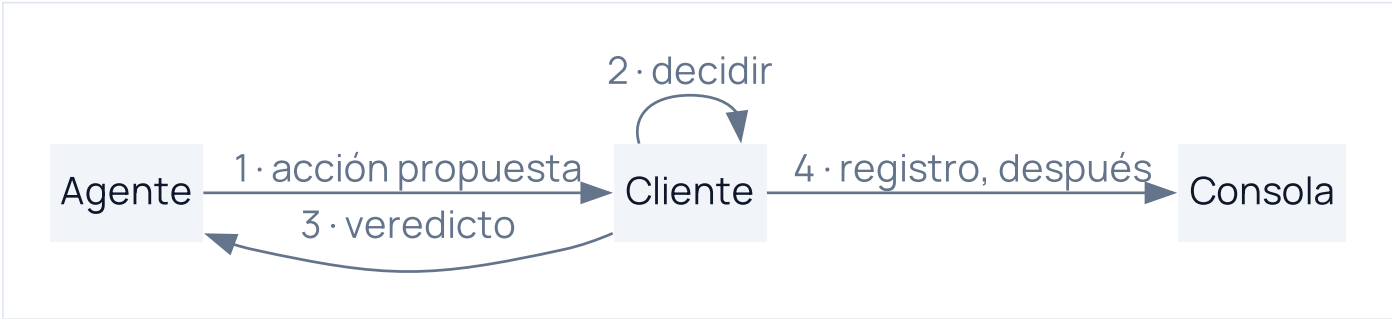


Figura 2. Una acción, un veredicto: la decisión se toma en la máquina antes de que la acción se ejecute, y el registro llega después.

Qué cruza la frontera de la máquina

Cuatro flujos cruzan la frontera de una máquina que ejecuta el cliente. Solo dos tienen que ver con OpenLatch.

Flujo	Sentido y contenido
Políticas	Descendente, de la consola a la máquina. El cliente comprueba al arrancar y a intervalos regulares si hay políticas más recientes que las que tiene
Registros	Ascendente, de la máquina a la consola, por HTTPS y a través de tu proxy, una vez decidida la acción
El tráfico de modelos de tus agentes	Sin cambios. Tus agentes llegan a sus puntos de acceso de modelos exactamente como antes, con sus propias credenciales, y OpenLatch no altera ese tráfico
Inicio de sesión en la consola	Hacia la consola desde un navegador, a través de tu proveedor de identidad

[Requisitos de red](#) enumera cada destino y cubre los proxies y la inspección TLS.

Cuando la red no está disponible

Una consola inaccesible es un problema de entrega, no de aplicación. Las políticas que ya están en la máquina siguen vigentes, y la máquina sigue decidiendo con ellas mientras dure el corte. Los registros que no se pueden enviar se guardan localmente, hasta un límite de tamaño, y suben cuando la consola vuelve a ser accesible. Un corte lo bastante largo para superar el límite descarta los registros no enviados más antiguos, y el cliente lo anota.

Las políticas no caducan solas. Una política que se desactivara tras un corte permitiría a cualquiera desactivar la aplicación desenchufando un cable, así que la máquina sigue aplicándolas y la consola muestra que las políticas de la máquina están desactualizadas. Una máquina que nunca ha recibido una política permite la acción y registra que no tenía política: OpenLatch nunca bloquea sin política. [Requisitos de red](#) recoge la tabla completa de estados degradados.

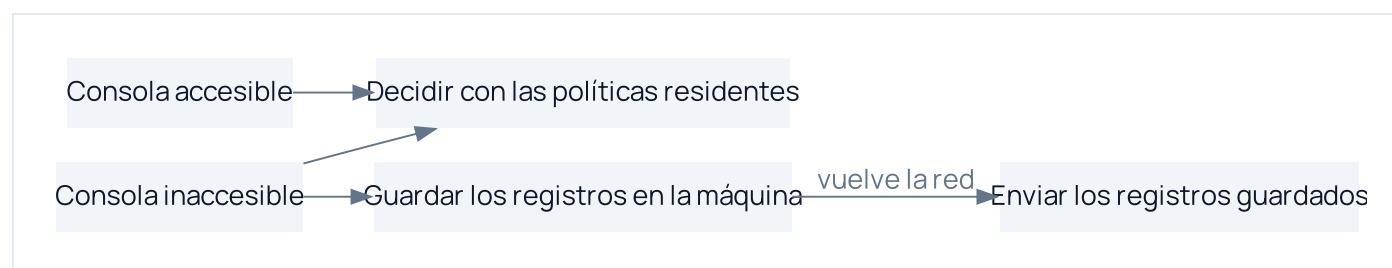


Figura 3. Decidir sin conexión: la máquina sigue decidiendo con las políticas que tiene, y los registros esperan a la red.

En resumen

- Un cliente por máquina que ejecuta un agente, a nivel de usuario, sin derechos de administrador. Solo escucha en el bucle local y no abre ningún puerto entrante.
- Guarda su configuración y sus registros locales en un único directorio, `~/ .openlatch` o `%APPDATA%\openlatch`, que `OPENLATCH_DIR` cambia de sitio.
- Las acciones se deciden en la máquina, antes de ejecutarse, frente a las políticas que la máquina ya tiene. No hay llamada de red en ese camino.
- La consola es un servicio, sin nada que instalar ni alojar. Contiene la organización, sus usuarios y roles, el registro de agentes y el registro de auditoría, y las políticas llegan a cada máquina desde ella.
- Una consola inaccesible no detiene la aplicación. Las políticas siguen vigentes, los registros se guardan y se envían después, y una máquina sin política permite la acción y registra por qué.

Requisitos previos

Completa esta página antes de programar la primera oleada de despliegue. Nada del despliegue es difícil de conseguir. Lo que frena un despliegue es descubrir, la mañana de la primera oleada, que una decisión está en otra parte de la organización, o que la persona que puede aprobar una entrada de tráfico saliente está ausente. El inventario siguiente convierte ese descubrimiento en un formulario que alguien completa la semana anterior.

Completa el inventario de preparación en una sola sesión de trabajo, con todas las personas que tienen una respuesta en la sala. La lista de comprobación convierte después esas respuestas en filas agrupadas por fase de trabajo. La última parte indica lo que cuesta el despliegue en horas.

Inventario de preparación

Cada respuesta cambia algo más adelante. Las respuestas sobre los agentes deciden qué máquinas van en qué oleada. Las respuestas sobre los equipos deciden el canal de instalación. Las respuestas de red deciden si la ruta se descubre o se proporciona. Las respuestas de identidad deciden quién administra la organización. Cuando la respuesta sea realmente «ninguno», escribe «ninguno» en lugar de dejarlo vacío, para que un lector posterior distinga una pregunta respondida de una que nunca se hizo.

Agentes

Elemento	Por qué importa
Qué agentes se usan y en qué máquinas	La instalación detecta los agentes presentes en una máquina y conecta cada uno. Conocer el conjunto de antemano indica qué máquinas van en qué oleada. En cuanto un agente envía datos, la consola muestra su nivel de control
Si algún agente se ejecuta fuera de un equipo personal	El cliente se instala una vez por máquina que ejecuta un agente, esté donde esté esa máquina. Un host de compilación o un equipo compartido que ejecuta un agente entra en el alcance igual que un portátil

Equipos

Elemento	Por qué importa
Combinación de sistemas operativos y número de máquinas por sistema	El cliente se instala y se ejecuta a nivel de usuario en cada plataforma compatible, sin derechos de administrador. Las versiones mínimas son macOS 13, Windows 10 1809 o Windows Server 2019, Ubuntu 20.04 y Debian 10
Arquitecturas de procesador del parque	Hay paquetes para macOS en Apple silicon e Intel, Linux x64 y arm64, y Windows x64. Windows en ARM64 y las distribuciones Linux basadas en musl, como Alpine, no son compatibles
El canal de distribución o de configuración que tu organización ya utiliza	Las oleadas 2 y 3 entregan la instalación por un canal que ya llega a estas máquinas. Ese canal también lleva la credencial de la oleada desde tu almacén de secretos
Si alguna máquina de una oleada la comparten varias personas	La instalación y el estado local son propios de cada usuario. Cada usuario de una máquina compartida hace su propia instalación

Red

Elemento	Por qué importa
Proxy en el camino: sí o no	Decide si la ruta se descubre en la instalación o se le proporciona. Consulta Requisitos de red
Tipo de proxy: HTTP CONNECT, SOCKS5 o configuración automática de proxy	Los tres son compatibles. El sistema operativo evalúa el archivo de configuración automática de proxy en Windows y macOS. En Linux, define la ruta explícitamente
Autenticación del proxy: ninguna, Basic, Kerberos (Negotiate) o NTLM	Ninguna, Basic y Negotiate son compatibles. NTLM no lo es, por diseño: un entorno solo NTLM prevé una ruta Negotiate o sin autenticación para el cliente antes de la primera oleada
Inspección TLS: sí o no, y dónde está la autoridad de certificación de interceptación	La inspección no requiere ninguna excepción. La raíz de interceptación va en el almacén de confianza del sistema operativo, o se proporciona como archivo PEM en la instalación
Si los puntos de acceso de modelos a los que ya llegan tus agentes siguen siendo accesibles y quedan exentos del almacenamiento en búfer de respuestas y de la inspección de flujos	Tus agentes llegan a esos puntos de acceso exactamente como antes. Un dispositivo que almacena en búfer una respuesta en streaming hace que la salida del agente llegue de golpe

Identidad

Elemento	Por qué importa
El proveedor de identidad SAML 2.0 de tu organización	La consola autentica una organización mediante SAML 2.0. Microsoft Entra ID, Okta, PingOne y Active Directory Federation Services se conectan así. Conectarlo requiere un administrador en cada lado
Dominios de correo permitidos	Los dominios permitidos y las invitaciones controlan juntos quién puede unirse a la organización
Directorio para buscar propietarios: Microsoft Entra ID, Okta, Google Workspace, PingOne o Active Directory	Una búsqueda de solo lectura asocia el propietario de un agente a una persona. Nunca crea ni elimina usuarios. Esta conexión es opcional
Quién tiene los roles owner y admin en la consola	Los roles de la consola son owner, admin, editor, viewer. Las personas invitadas empiezan como viewer, y los ajustes de identidad requieren admin

Secretos

Elemento	Por qué importa
Dónde se guarda la OPENLATCH_API_KEY de cada oleada y cómo llega a una máquina sin aparecer en un script	Una clave por oleada de despliegue, creada en la consola y revocada allí cuando la oleada termina. Las instalaciones gestionadas la leen de tu almacén de secretos como variable de entorno. Nunca se escribe en un script, un paquete ni una línea de comandos

Piloto

Elemento	Por qué importa
Las diez máquinas de la oleada 1 y sus propietarios	La oleada 1 sigue la ruta de desarrollador. Los propietarios con nombre permiten cotejar cada máquina con el registro de agentes de la consola
Cómo contactarás con OpenLatch durante el despliegue	Acuerda la vía de soporte antes de la primera oleada, no durante. Consulta Operar

Nota

Un entorno cuyo proxy es solo NTLM es la única respuesta de este inventario que puede bloquear un despliegue de entrada. Pregunta pronto al equipo que gestiona tu proxy, y lee la matriz de compatibilidad de proxies en [Requisitos de red](#) antes de programar la primera oleada.

Lista de comprobación consolidada

Cada fila indica de dónde viene el valor y la página que bloquea. Las cuatro fases son independientes: las filas de la consola y las de red avanzan en paralelo la misma semana.

Preparar la consola

#	Elemento	De dónde viene	Bloquea
1	Organización creada en la consola	Tu contacto de OpenLatch crea la organización e invita a su propietario. Acepta la invitación desde la dirección invitada. Los inicios de sesión posteriores empiezan en <code>app.openlatch.ai</code> con un paso de correo cuya comprobación en el navegador necesita <code>challenges.cloudflare.com</code>	Configurar la organización
2	SAML conectado y probado	La consola de administración de tu proveedor de identidad, para el identificador de entidad, la URL de inicio de sesión y el certificado o la URL de metadatos. Se introducen en la consola en Settings > SAML SSO	Configurar la organización
3	Dominios de correo permitidos definidos	Los mismos ajustes SAML que la fila 2	Configurar la organización
4	Roles owner, admin, editor y viewer asignados	La consola, en Settings > Users . Las personas invitadas empiezan como viewer	Configurar la organización
5	Directorio elegido para buscar propietarios, opcional	La consola de administración de tu directorio. La búsqueda es de solo lectura	Configurar la organización
6	Conservación del registro de auditoría definida	La consola, en Settings > Audit Trail : 90, 180, 365 o 730 días, o un periodo personalizado, con purga automática y exportación	Configurar la organización
7	Canal de notificación por webhook o Slack, opcional	La consola, en Settings > Notification Channels	Configurar la organización
8	Credencial de la oleada creada y guardada	La consola, en API Keys de tu cuenta, desde una sesión de navegador iniciada. Guardada en el almacén de secretos de tu organización	Configurar la organización

Preparar la red

#	Elemento	De dónde viene	Bloquea
9	Entrada de tráfico saliente para <code>app.openlatch.ai</code> en TCP 443 aprobada	Tu proceso de cambios, según la tabla de Requisitos de red	Desplegar el cliente
10	Los puntos de acceso de modelos a los que ya llegan tus agentes siguen siendo accesibles y quedan exentos del almacenamiento en búfer de respuestas y de la inspección de flujos	La configuración de tu proxy o de tu pasarela web segura	Desplegar el cliente
11	Tipo de proxy y vía de credenciales confirmados	La configuración de tu proxy, leída frente a la matriz de compatibilidad	Desplegar el cliente
12	Autoridad de certificación de interceptación disponible como archivo PEM, si el sistema operativo aún no confía en ella	La exportación de tu propia autoridad de certificación	Desplegar el cliente

Preparar las máquinas

#	Elemento	De dónde viene	Bloquea
13	Los agentes que quieres cubrir están instalados y con sesión iniciada en esas máquinas	Las propias máquinas, confirmadas con la persona que usa cada una	Desplegar el cliente
14	Instalación revisada para la ruta gestionada	La ruta gestionada en Desplegar el cliente . Lee <code>OPENLATCH_API_KEY</code> de tu almacén de secretos y nunca la lleva en claro	Desplegar el cliente

Preparar el despliegue

#	Elemento	De dónde viene	Bloquea
15	Lista de máquinas de la oleada 1 con sus propietarios	Tu inventario de equipos, cotejado con los propietarios de los agentes	Verificar y desplegar
16	Desarrolladores informados de que algunos agentes les piden confirmar la conexión una vez	Tu propia comunicación del despliegue, enviada antes de que empiece la oleada	Verificar y desplegar
17	Vía de soporte acordada para el despliegue	Operar	Verificar y desplegar
18	Responsable de la marcha atrás designado	Tu propia decisión: la persona que puede pausar una oleada y retirar el cliente de una máquina	Verificar y desplegar

Esfuerzo y calendario

El despliegue del cliente se hace en tres oleadas de dos semanas. La oleada 1 instala unas diez máquinas por la ruta de desarrollador. La oleada 2 lleva unas cincuenta máquinas por el canal que ya utilizas. La oleada 3 se extiende al perímetro completo y termina con la decisión de pasarlo a operación habitual.

El esfuerzo se concentra en la primera semana, cuando se prepara la organización y se demuestra la ruta en una máquina de cada sistema operativo del parque. Después, cada oleada repite un procedimiento que ya funcionó.

Fase de trabajo	Oleada 1	Oleada 2	Oleada 3	Esfuerzo habitual
Preparar la consola	Conectar SAML, definir los dominios permitidos y la conservación de la auditoría, invitar a las personas, crear la credencial de la oleada	Renovar la credencial de la oleada	Traspasar la rutina de operación	Un día la primera semana, luego unas horas por semana
Preparar la red	Aprobar la entrada de tráfico saliente y confirmar la ruta del proxy	Revisar la ruta gestionada	Ningún cambio más	Media jornada la primera semana
Preparar las máquinas	Demostrar la instalación en una máquina por sistema operativo y pasar diez máquinas por la ruta de desarrollador	Llevar la instalación al canal que ya utilizas y lanzar la oleada	Extender el mismo canal a las máquinas restantes	Unos minutos por máquina tras la primera
Confirmar y operar	Confirmar que cada máquina instalada aparece en el registro de agentes con un propietario	Revisión semanal y revisión de la ruta gestionada	Revisión semanal y, después, la decisión de pasar a operación habitual	Una revisión semanal durante todo el despliegue

En resumen

- Completa el inventario de preparación en una sola sesión antes de programar la primera oleada. Cada casilla vacía es una decisión que alguien tomará más tarde con prisa.
- Una única entrada de tráfico saliente es obligatoria: `app.openlatch.ai` en TCP 443, saliente. Todo lo demás en el camino es el tráfico propio de tus agentes, que no cambia.
- Un entorno cuyo proxy es solo NTLM bloquea el despliegue. Prevé una ruta Negotiate o sin autenticación para el cliente.

- No se necesitan derechos de administrador en una máquina. La instalación y el estado local son propios de cada usuario: una máquina compartida por varias personas necesita una instalación por usuario.
- Una `OPENLATCH_API_KEY` por oleada de despliegue, guardada en tu almacén de secretos y revocada en la consola cuando la oleada termina. Nunca aparece en un script, un paquete ni una línea de comandos.

Requisitos de red

Usa esta página como solicitud de cambio para tu equipo de red. El cliente necesita un único destino saliente, alcanzado a través de tu proxy, y no abre nada de entrada.

Una única dependencia saliente

El cliente se ejecuta en la máquina que ejecuta el agente, dentro del propio ciclo de vida del agente. Contacta con OpenLatch para tres cosas: obtener las políticas con las que decide, entregar el registro de lo que decidió y autenticarse. Las tres van a un único destino por un único puerto. La telemetría de producto y los informes de fallos, opcionales, van a ese mismo destino, así que OpenLatch no añade ningún otro destino, tanto si los dejas activos como si los desactivas.

Propiedad	Valor
Sentido	Solo saliente. El cliente no abre ningún puerto de escucha en una interfaz de red, y nada de la máquina pasa a ser accesible desde la red
Destino	<code>app.openlatch.ai</code> , en TCP 443
Transporte	HTTPS. TLS se verifica con el almacén de confianza del sistema operativo, más un paquete de certificados opcional combinado encima. La verificación no se puede desactivar
A través de tu proxy	Sí: HTTP y HTTPS CONNECT, SOCKS5 y configuración automática de proxy donde el sistema operativo la admite, con autenticación Basic y Kerberos (Negotiate)
Lista de IP permitidas	No compatible. No hay ningún rango de direcciones estático: permite por nombre de dominio completo y por SNI
Certificados	OpenLatch no emite ninguno ni te exige ninguno



Figura 1. Un parque de máquinas detrás de un proxy corporativo: un único destino saliente, nada de entrada.

Tráfico saliente requerido

Esta es la tabla que hay que pegar en una solicitud de cambio. La primera fila es lo que OpenLatch necesita para funcionar, y es el único destino que una máquina que ejecuta el cliente necesita de OpenLatch. La segunda solo es necesaria mientras instalas o actualizas el cliente; un entorno con un espejo interno de paquetes la apunta a su espejo. La tercera solo es necesaria para instalar con uno de los dos scripts. La cuarta solo afecta a los navegadores.

Destino	Puerto	Sentido	Protocolo	Uso	Cómo permitirlo
<code>app.openlatch.ai</code>	TCP 443	Saliente	HTTPS	Políticas, registros de decisiones, autenticación, y la telemetría de producto y los informes de fallos opcionales	Por nombre y SNI. No hay ningún rango de direcciones estático
Tu registro de paquetes: <code>registry.npmjs.org</code> , salvo que apuntes el cliente a un espejo	TCP 443	Saliente	HTTPS	Instalar el cliente y descargar una actualización cuando la permitas	Por nombre, o por el nombre de tu espejo interno. Solo necesario durante una instalación o una actualización
<code>openlatch.ai</code>	TCP 443	Saliente	HTTPS	Descargar el script de instalación, <code>install.sh</code> o <code>install.ps1</code> , y el informe de instalación que envía	Por nombre. Solo necesario al instalar con un script
<code>challenges.cloudflare.com</code> , desde el navegador de cualquier persona que inicie sesión en la consola	TCP 443	Saliente	HTTPS	La comprobación que hace el paso de correo	Por nombre, solo para navegadores. Ninguna máquina que

Destino	Puerto	Sentido	Protocolo	Uso	Cómo permitirlo
				al iniciar sesión	ejecuta el cliente lo contacta

Nota

En un entorno que deniega el tráfico saliente por defecto, la segunda y la tercera filas son las que se olvidan. La instalación falla entonces antes de que el cliente haya llegado nunca a la consola, y el fallo parece un producto roto más que una ruta cerrada.

El navegador del administrador

El inicio de sesión en la consola empieza con un paso de correo, y ese paso hace en el navegador una breve comprobación de que el visitante es una persona. La comprobación se sirve desde `challenges.cloudflare.com`, así que los navegadores con los que inician sesión tus administradores deben llegar a ese nombre en TCP 443. Un navegador que no puede llegar falla la comprobación cada vez. Este requisito solo afecta a los navegadores: `app.openlatch.ai` sigue siendo el único destino que necesita una máquina que ejecuta el cliente.

El tráfico propio de tus agentes

Tus agentes llegan a sus puntos de acceso de modelos exactamente como antes de instalar OpenLatch. OpenLatch no termina ese tráfico ni lo vuelve a cifrar con otro certificado, y no le añade ningún destino propio. Esos puntos de acceso deben seguir siendo accesibles y quedar exentos del almacenamiento en búfer de respuestas y de la inspección de flujos, para que una respuesta en streaming no se retenga de camino a la persona que la aguarda.

Un dispositivo que almacena la respuesta en búfer, o la analiza como un cuerpo completo, retiene el flujo hasta que termina y luego lo libera de golpe. La petición sigue teniendo éxito; el desarrollador ve un agente que parece atascarse y luego lo imprime todo de una vez.

Para los productos de proxy que dirigen el tráfico por aplicación en lugar de por destino, el nombre del proceso es `openlatch` en macOS y Linux y `openlatch.exe` en Windows. Usa ese nombre en una definición de aplicación personalizada.

Matriz de compatibilidad de proxies

Responde control por control. Una fila «No» es una decisión de diseño más que una carencia, y cada una indica la vía alternativa.

Control de red	Compatible	Cómo se gestiona	Notas
Proxy HTTP o HTTPS CONNECT sin autenticación	Sí	Se descubre en la instalación, o se le proporciona a la instalación	La forma habitual en las empresas. La ruta que funciona se registra en la máquina
Autenticación Basic	Sí	La credencial se pide ante un desafío y se guarda en el almacén de credenciales del sistema operativo	Nunca se escribe en texto claro en un archivo
Kerberos (Negotiate)	Sí	Usa la identidad de la sesión iniciada, mediante el mecanismo del sistema operativo	El principal de servicio es por defecto el servicio HTTP del host del proxy y se puede sustituir en la instalación. Un ticket ausente o caducado se notifica con su solución
NTLM	No	No compatible, por diseño	Microsoft dejó obsoleto NTLM, y un cliente que retrocede a él discretamente va en contra de aquello para lo que se desplegó. Usa Negotiate, o una ruta sin autenticación para el cliente
Configuración automática de proxy (PAC o WPAD)	Sí, en Windows y macOS	La URL se entrega al resolvidor del sistema operativo	Se rechaza en Linux, donde el cliente no dispone de un resolvidor de configuración automática. Define la ruta explícitamente allí
SOCKS5	Sí	Una ruta socks5: // o socks5h: //	La resolución de nombres sigue siendo local con socks5 y pasa al proxy con socks5h
Proxy transparente	Sí	Sin configuración en la máquina; el tráfico se intercepta en el camino	Combínalo con la fila de inspección TLS si el dispositivo vuelve a firmar la conexión

Control de red	Compatible	Cómo se gestiona	Notas
Inspección TLS con una autoridad de certificación corporativa	Sí	La raíz de interceptación va en el almacén de confianza del sistema operativo, o se proporciona como paquete PEM en la instalación	El paquete se combina encima de las raíces del sistema, nunca las sustituye, y la verificación nunca se desactiva
Certificados de cliente mTLS hacia el proxy	No	—	El cliente no presenta ningún certificado de cliente. Usa uno de los esquemas de autenticación compatibles
Lista de IP permitidas para el tráfico saliente de OpenLatch	No	—	No hay ningún rango de direcciones estático. Permite <code>app.openlatch.ai</code> por nombre y SNI

Configurar la ruta

Cómo se encuentra la ruta

En la mayoría de las máquinas corporativas, la instalación encuentra la ruta sin que se le indique. Sondea lo que ya está configurado y luego recorre los ajustes del sistema operativo, incluida la configuración automática de proxy en Windows y macOS. Solo pregunta si nada de eso llega a la consola, una vez, en un terminal. Una instalación desatendida que no encuentra ruta se detiene con un informe que nombra cada candidato que probó, en lugar de salir en directo sin avisar.

La ruta que permitió la instalación queda registrada en la máquina. Esto importa: el cliente arranca con la sesión del usuario y no hereda las variables del shell de instalación, así que una ruta que solo existía en una variable de entorno se perdería en el siguiente reinicio.

Proporcionar la ruta a la instalación

Cuando el descubrimiento no puede funcionar, o una instalación gestionada se ejecuta sin terminal en el que preguntar, proporciona la ruta directamente a la instalación. Proporciona a la vez el paquete de

certificados si el sistema operativo aún no confía en la raíz de interceptación.

Retén la inscripción con `OPENLATCH_NO_INIT` y ejecútala en un segundo paso, para que las opciones lleguen al comando que la realiza. Exporta la variable antes de que se ejecute el script, para que el shell que lee el script también la lleve.

En macOS y Linux:

```
export OPENLATCH_NO_INIT=1
curl -LsSf https://openlatch.ai/install.sh | sh
~/local/bin/openlatch init --proxy http://proxy.corp:8080 --ca-bundle /etc/ssl/corp-root.pem
```

En Windows:

```
$env:OPENLATCH_NO_INIT = "1"
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
& "$env:LOCALAPPDATA\openlatch\bin\openlatch.exe" init --proxy http://proxy.corp:8080 --ca-bundle C:\corp\corp-root.pem
```

En una máquina que ya tiene Node.js, `npx @openlatch/client init` acepta las mismas opciones en un solo paso:

```
npx @openlatch/client init --proxy http://proxy.corp:8080 --ca-bundle /etc/ssl/corp-root.pem
```

Se puede proporcionar una lista de exclusiones junto con la ruta, con la sintaxis habitual. El tráfico que no sale de la máquina no pasa por el proxy y no necesita exclusión.

Cuando la política prohíbe que una máquina salga sin el proxy, el cliente se puede configurar para rechazar una conexión directa si no se resuelve ninguna ruta. Entonces notifica un fallo de ruta en lugar de saltarse tus controles sin avisar. [Configuración del cliente](#) enumera los ajustes.

Gestión de credenciales

Una credencial de proxy se guarda en el almacén de credenciales del sistema operativo: el Administrador de credenciales en Windows, el Llavero en macOS, Secret Service en Linux, y un archivo cifrado en un host sin interfaz que no tenga ninguno. Nunca se escribe en texto claro en un archivo de configuración. En todos los lugares donde se muestra o se registra una URL de proxy, aparece como `user : *****@host`. Se rechaza una URL de proxy que lleve una contraseña en la línea de comandos, porque cualquier proceso de la máquina puede leer una línea de comandos.

Inspección TLS

La conexión con la consola funciona bajo inspección y no necesita ninguna excepción. El cliente hace llamadas HTTPS normales de petición y respuesta, verifica con el almacén de confianza del sistema operativo y no fija ningún certificado. Coloca la raíz de interceptación en el almacén de confianza donde tu entorno ya lo hace, o proporciona una copia PEM en la instalación. El paquete se combina encima de las raíces del sistema, así que una máquina sigue confiando en todo aquello en lo que ya confiaba.

Los puntos de acceso de modelos de tus agentes son la excepción, y la petición ahí es acotada: exímelos del almacenamiento en búfer de respuestas y de la inspección de flujos, y deja intacto el resto de tu política de inspección.



Precaución

Nunca desactives la verificación de certificados para sortear un proxy que inspecciona. El cliente no ofrece ningún ajuste para ello. Un fallo de negociación se notifica con su solución: confiar en la raíz de interceptación en la máquina, o proporcionarla en la instalación.

Comportamiento sin conexión y degradado

El cliente actualiza sus políticas desde la consola en segundo plano. Cada decisión se toma en la máquina, con las políticas que ya están allí, así que ninguna llamada de red está en el camino de decisión. Eso es lo que permite que un veredicto sobreviva a una ruta cortada.

Donde la máquina tiene políticas, sigue aplicándolas haga lo que haga la red. Donde no tiene ninguna, permite la acción y marca el registro para indicar que la decisión se tomó sin política. Nada falla en silencio.

Situación	Qué hace la máquina	Qué ves
La consola no es accesible	Sigue aplicando las políticas que tiene y reintenta en segundo plano. Los registros se guardan en la máquina	La máquina deja de comunicarse. <code>openlatch status</code> la muestra desconectada, con políticas presentes
Las políticas de la máquina no se actualizan desde hace tiempo	Sigue aplicándolas. Las políticas no caducan	Un aviso de desactualización. La desactualización se notifica, nunca se actúa en consecuencia
La credencial de la oleada se revoca o se renueva, y la consola la rechaza	Deja de obtener actualizaciones, sigue aplicando lo que tiene y se reanuda sola en cuanto hay una credencial válida	Un aviso de autenticación en la máquina y ninguna comunicación más en la consola
Tu organización aún no ha publicado ninguna política	Mantiene vigente lo que ya tiene	La máquina figura como conectada, sin nada que obtener
Llega una actualización dañada o en una forma que la máquina no reconoce	La rechaza y conserva las políticas anteriores. Una actualización dañada nunca entra en vigor	Un aviso de integridad en la máquina, con las políticas anteriores aún vigentes
La copia local de las políticas se modifica en la máquina	La descarta. La máquina funciona sin política hasta la siguiente actualización correcta	Cada veredicto se marca como decidido sin política, y la máquina notifica la pérdida
Ninguna política ha llegado nunca a la máquina	Devuelve Allow: una máquina nunca bloquea sin política	Cada veredicto se marca como decidido sin política
El cliente no se está ejecutando	La acción continúa y no se retiene	<code>openlatch status</code> indica que el cliente no se ejecuta, y la consola no muestra actividad reciente de la máquina

Situación	Qué hace la máquina	Qué ves
Los registros no se pueden entregar	Reintenta, luego los guarda en la máquina y los envía cuando la consola vuelve a ser accesible	Un hueco en la consola que se cierra cuando la máquina se reconecta. La reserva está acotada: por encima de su límite de tamaño se descartan los registros no enviados más antiguos, y el registro del daemon anota un aviso 0L-1204
Pasas todas las políticas a Monitor desde la consola	Cada máquina aplica el cambio en su siguiente actualización y registra lo que habría hecho	El cambio, en la consola, sin necesidad de acceder a ninguna máquina. Las decisiones se siguen registrando

Verificar la ruta

Ejecuta estos comandos en la primera máquina antes de la oleada, y de nuevo en cualquier máquina que notifique un problema de conexión.

```

openlatch status
openlatch doctor
openlatch system proxy test

```

Comando	Qué demuestra
openlatch status	Si la máquina está conectada a la consola, si hay políticas presentes y qué agentes están conectados. Responde incluso cuando la ruta está cortada, que es justo cuando lo necesitas
openlatch doctor	La máquina completa, con una causa y una solución para cada problema detectado. Informa del salto por tu proxy por separado de la conexión con la consola, para que un proxy caído y una consola inaccesible nunca parezcan el mismo fallo
openlatch system proxy test	La propia ruta, sondeada salto a salto. Se detiene en el salto que falla y lo nombra

[Desplegar el cliente](#) cubre su instalación, y [Verificar y desplegar](#) la verificación completa de una máquina.

En resumen

- Aprueba una única entrada para las máquinas: `app.openlatch.ai` en TCP 443, saliente. Sin puertos entrantes.
- Los navegadores que inician sesión en la consola también necesitan `challenges.cloudflare.com` en TCP 443. Ninguna máquina que ejecuta el cliente lo contacta.
- Permite por nombre y SNI, no por dirección. No hay ningún rango de direcciones estático.
- Tus agentes llegan a sus puntos de acceso de modelos como antes. Mantén esos puntos de acceso accesibles y exentos del almacenamiento en búfer de respuestas y de la inspección de flujos.
- La inspección TLS no necesita ninguna excepción para la conexión con la consola. Los proxies Basic, Kerberos, SOCKS5, transparentes y de configuración automática son compatibles. NTLM no lo es: prevé una ruta Negotiate o sin autenticación.
- Una ruta cortada degrada la entrega, no la aplicación. La máquina sigue decidiendo con las políticas que tiene, los registros esperan a que la consola sea accesible, y ambos estados aparecen en `openlatch status`.

Configurar la organización

Prepara la consola para tu organización: la cuenta, el inicio de sesión único, los usuarios y sus roles, la búsqueda de propietarios, la conservación y las notificaciones, y la credencial con la que se inscribe el cliente. Todo ocurre en el navegador, lleva una media hora y no toca ninguna máquina. La instalación del cliente está en [Desplegar el cliente](#).

Antes de empezar, ten completado el [inventario de preparación](#) y comprueba que tu navegador llega a `challenges.cloudflare.com`.

Iniciar sesión y completar la organización

OpenLatch crea la organización por ti, designa como propietaria la dirección que diste a tu contacto de OpenLatch y envía una invitación a esa dirección. El asistente de bienvenida es donde terminas de configurarla.

- 1 Acepta la invitación.** Abre el enlace del correo de invitación. Te lleva a la página de inicio de sesión propia de tu organización y admite la dirección invitada como propietaria. Complétalo con cualquier método que ofrezca la página: una passkey, un enlace mágico, una dirección de correo y una contraseña, o un inicio de sesión social.

Resultado esperado: la consola se abre en tu organización y tu rol indica owner.

- 2 Inicia sesión desde `app.openlatch.ai`.** Cada inicio de sesión posterior empieza con un paso: tu dirección de correo. Ese paso hace en tu navegador una breve comprobación de que el visitante es una persona; normalmente no pide nada y, como mucho, un clic. La consola encuentra después la organización a la que pertenece tu dirección y abre su página de inicio de sesión con la dirección ya rellena.

A una dirección que ninguna organización conoce se le informa de ello. Una comprobación que falla cada vez suele significar que el navegador no llega a `challenges.cloudflare.com`; consulta [Requisitos de red](#).

- 3 Recorre el asistente de bienvenida.** Tiene cuatro pasos: nombrar la organización, invitar a compañeros, conectar un agente y un resumen de lo que completaste y lo que omitiste. Usa el primero

para definir la razón social o el nombre de la unidad de negocio que quieres ver en el registro de auditoría. Conectar un agente está en [Desplegar el cliente](#); omitirlo aquí no cuesta nada.

Resultado esperado: el asistente se cierra en la consola, sin agentes inscritos todavía.

Conectar el inicio de sesión único

El inicio de sesión único usa SAML 2.0, configurado por organización. Microsoft Entra ID, Okta, PingOne y Active Directory Federation Services se conectan todos por la misma pantalla. Los ajustes de identidad requieren el rol admin o superior.

1 Abre Settings > SAML SSO y añade un proveedor. La página muestra los datos del proveedor de servicios que necesitará quien administre tu proveedor de identidad. Ten a mano los metadatos de tu proveedor de identidad.

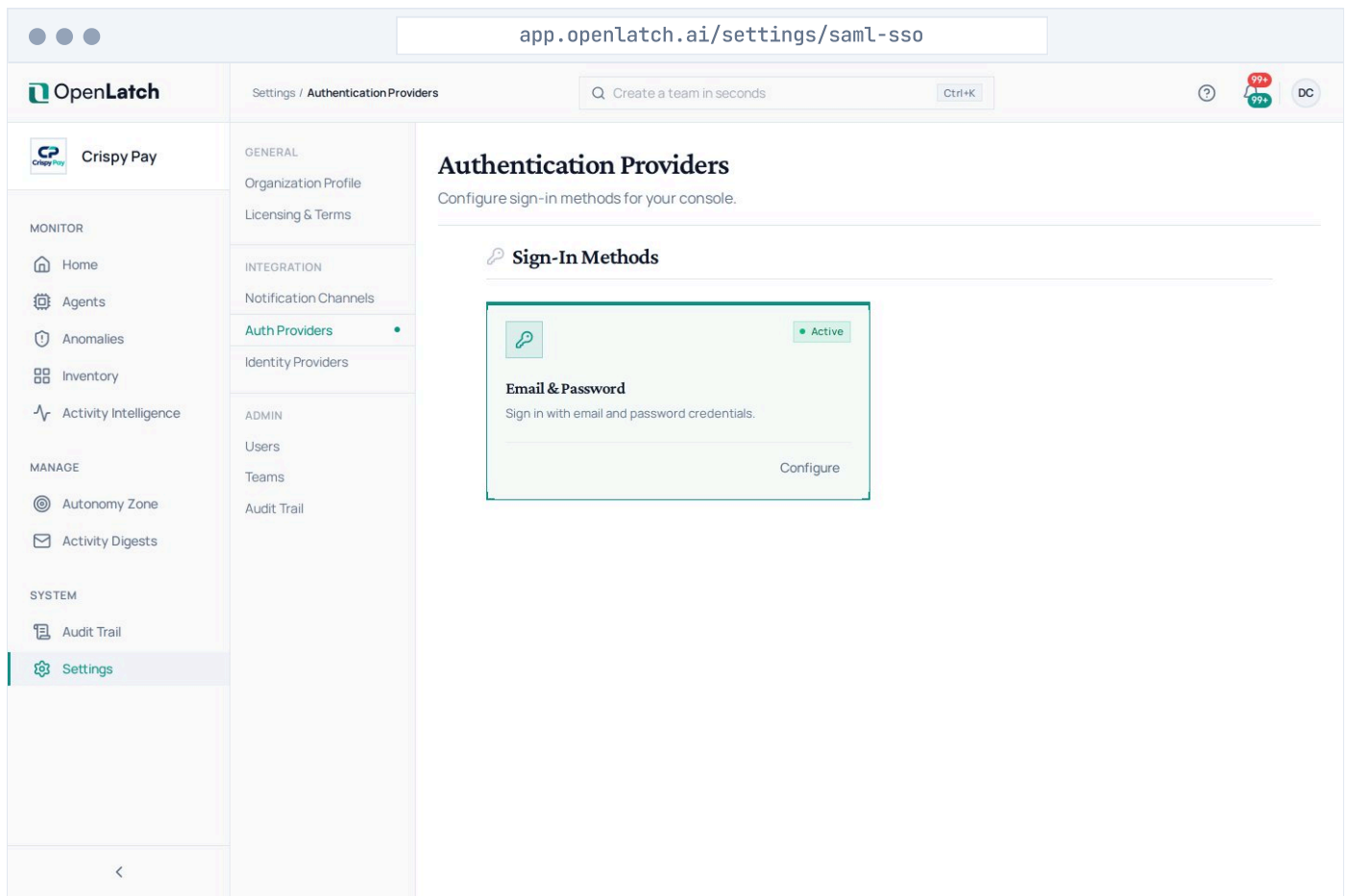
2 Introduce los datos del proveedor de identidad. La consola pide el identificador de entidad del proveedor de identidad, su URL de inicio de sesión, su certificado de firma y los dominios de correo autorizados a iniciar sesión. Puedes proporcionar una URL de metadatos en lugar de pegar el certificado. Elige primero tu proveedor de identidad en la lista y la página muestra una guía de configuración escrita para ese producto.

La aserción que envía tu proveedor de identidad debe incluir la dirección de correo y el nombre visible del usuario; la guía de configuración nombra los campos exactos.

3 Guarda, ejecuta la prueba de conexión y luego inicia sesión. La prueba se ejecuta con la configuración guardada, no con lo que aún está en pantalla. Comprueba que la URL de metadatos o la URL de inicio de sesión del proveedor de identidad responde; no realiza un inicio de sesión SAML. Después inicia sesión tú mismo por SAML, en una ventana privada del navegador, antes de pedirselo a nadie más.

Resultado esperado: la prueba se supera, y tu propio inicio de sesión SAML llega a la consola.

4 Decide quién puede unirse. Dos puertas controlan la pertenencia. Un dominio de correo permitido deja que cualquiera con una dirección de ese dominio se registre en la organización. Una invitación admite a una persona concreta. Mantén los dominios permitidos acotados y recurre a las invitaciones para todos los demás.



Los ajustes SAML SSO, con los datos del proveedor de servicios y los valores que proporciona tu proveedor de identidad.

[Configurar el inicio de sesión único](#) describe con más detalle los ajustes SAML y los demás métodos de inicio de sesión.

Invitar al equipo y asignar roles

Cuatro roles siguen un orden estricto de privilegios. Cada persona invitada empieza como viewer, así que un rol se concede deliberadamente en lugar de retirarse después.

Rol	Qué permite
owner	Todo lo que permite admin, más la propiedad de la propia organización. Déjalo en manos de la persona responsable del despliegue
admin	Ajustes de identidad, políticas, credenciales del cliente, conservación y canales de notificación. Concédelo a las pocas personas que llevan el despliegue
editor	Políticas y sustituciones del propietario de un agente
viewer	Lee los agentes, las ejecuciones, las decisiones y el registro de auditoría. Suficiente para todos los demás, incluido un desarrollador que comprueba que su máquina se inscribió

1 Invita por dirección de correo en **Settings > Users.** Cada invitación genera un enlace de registro ligado a esa dirección. Invita a las personas que llevan el despliegue, no a toda la organización de ingeniería.

Resultado esperado: cada persona invitada aparece en la lista de usuarios como pendiente, con el rol viewer.

2 Sube los roles que haga falta. Concede admin solo donde haya que cambiar ajustes de identidad, credenciales o conservación. Concede editor a quien mantiene las políticas y el registro de agentes. Deja al resto como viewer.

app.openlatch.ai/settings/users

OpenLatch

Settings / Users

Q Search by name or email... Ctrl+K

99+ 99+ DC

Users Invite User

Manage team members and invitations.

Q Search by name or email... + Status + Team + Email Created Last Login

☐	NAME ↕	EMAIL ↕	TEAMS	ROLE	STATUS ↕	LAST LOGIN ↕	CREATED ↕
☐	DO Docs Capture Owner	docs-cap...	Administr...	Owner	Active		
☐	RK Rohan Kapoor	rohan.ka...	Policy Edi...	Editor	Active		
☐	CL Camille Lefèvre	camille.le...	Policy Edi...	Editor	Active		
☐	MW Marcus Washington	marcus...	Policy Edi...	Editor	Active		
☐	IS Ingrid Solberg	ingrid.sol...	Administr...	Admin	Active		
☐	UD User Documentation	user-doc...	Administr...	Admin	Active		
☐	Luc Delsalle	luc.delsal...	Administr...	Owner	Active		

Showing 1 to 7 of 7 users

Los ajustes Users: cada miembro y el rol que tiene.

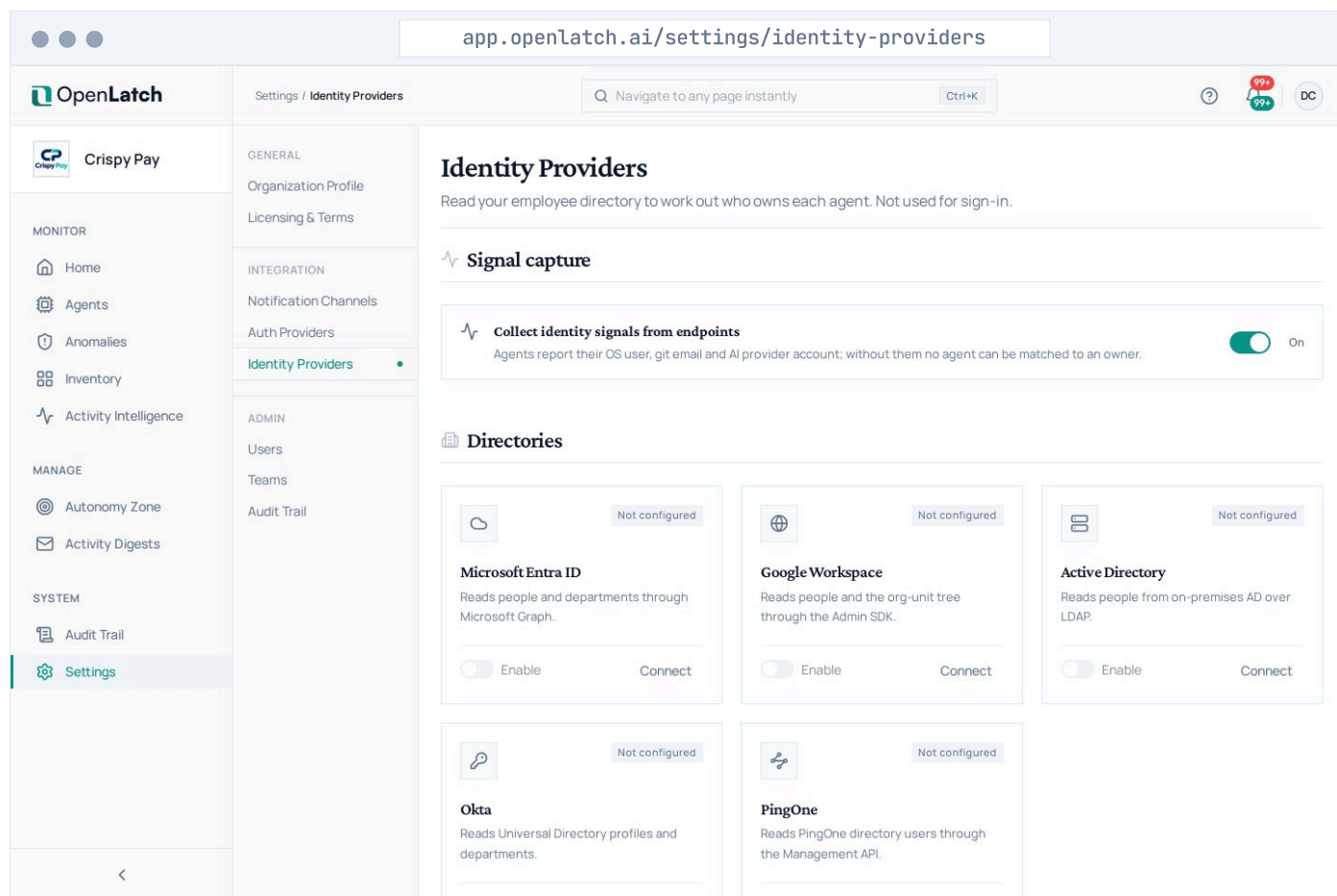
Conectar un directorio para buscar propietarios

Este paso es opcional y puedes hacerlo después de inscribir las primeras máquinas. Una conexión de directorio determina el propietario de cada agente conectado, para que el registro de agentes muestre una persona en lugar del nombre de una cuenta del sistema operativo. Es de solo lectura y no se usa para iniciar sesión.

La propiedad se determina desde dos lados, y los dos deben estar presentes. Las máquinas envían las señales que identifican quién trabaja, y el directorio convierte esas señales en una persona con nombre. El lado de las señales es un interruptor en la misma página, **Collect identity signals from endpoints**, activado por defecto. Si está desactivado, ningún agente puede asociarse a un propietario.

- 1 Abre **Settings > Identity Providers** y conecta un directorio: Microsoft Entra ID, Google Workspace, Active Directory, Okta o PingOne. Cada uno es una fuente de solo lectura; ninguno crea usuarios y ninguno copia el directorio en OpenLatch.

- Busca a una persona que sepas que está en el directorio. Una organización sin directorio conectado es un estado admitido, y un agente cuyo propietario no puede determinarse se registra como sin atribuir en lugar de suponerse.



Identity Providers: el interruptor que permite a las máquinas indicar quién trabaja, y los directorios que lo convierten en una persona con nombre.

Definir la conservación y las notificaciones

Define estos dos ajustes antes de empezar el despliegue; una revisión de seguridad preguntará por ambos.

- Define la conservación del registro de auditoría** en **Settings > Audit Trail**. El registro de auditoría es de solo anexo y está encadenado por hash. Elige un periodo de conservación, decide sobre la purga automática y define el formato de exportación que espera tu proceso de evidencias.

Ajuste	Valores
Conservación	90, 180, 365 o 730 días, o un periodo personalizado. La consola recomienda 365
Purga automática	Activada o no. Si está activada, los registros que superan el periodo de conservación se eliminan cada 24 horas
Exportación	CSV o JSON

- Añade un canal de notificación** en **Settings > Notification Channels**. Apúntalo al canal en el que ya trabaja el equipo del despliegue, como un canal de Slack o un webhook HTTPS, y envía una prueba.

Resultado esperado: llega el envío de prueba.

The screenshot shows the OpenLatch web interface at the URL `app.openlatch.ai/settings/audit-trail`. The left sidebar contains navigation links for Monitor, Manage, and System. The main content area is titled "Audit Trail" and includes the following sections:

- Retention Policy:** Includes a "Retention Period" section with buttons for 90 days, 180 days, 365 days (recommended), 730 days, and Custom. Below it is an "Auto-Purge" toggle switch which is currently turned on.
- Export Preferences:** Includes a section to "Choose the default format for audit trail exports" with two radio buttons: "CSV – Spreadsheet-compatible, human-readable" (selected) and "JSON – Machine-readable, preserves nested data".
- Hash Chain Integrity:** Includes a "Verify Integrity" button and a description: "Verify the cryptographic hash chain that ensures audit events have not been tampered with."

At the bottom right of the page, there are "Cancel" and "Save Configuration" buttons.

Los ajustes Audit Trail: periodo de conservación, purga automática y formato de exportación.

Crear la credencial del cliente

El cliente se autentica ante la consola con una clave de API. La creas desde una sesión de navegador iniciada, en la página **API Keys** de tu cuenta, y la revocas desde la misma página.

- 1 **Crea una clave por oleada de despliegue**, con el nombre de su oleada. Una clave por oleada significa que una clave que haya que revocar solo afecta a una oleada.

Resultado esperado: el valor de la clave se muestra una sola vez, completo, al crearla. Después, la lista muestra su nombre, un valor enmascarado, cuándo se creó y cuándo se usó por última vez.

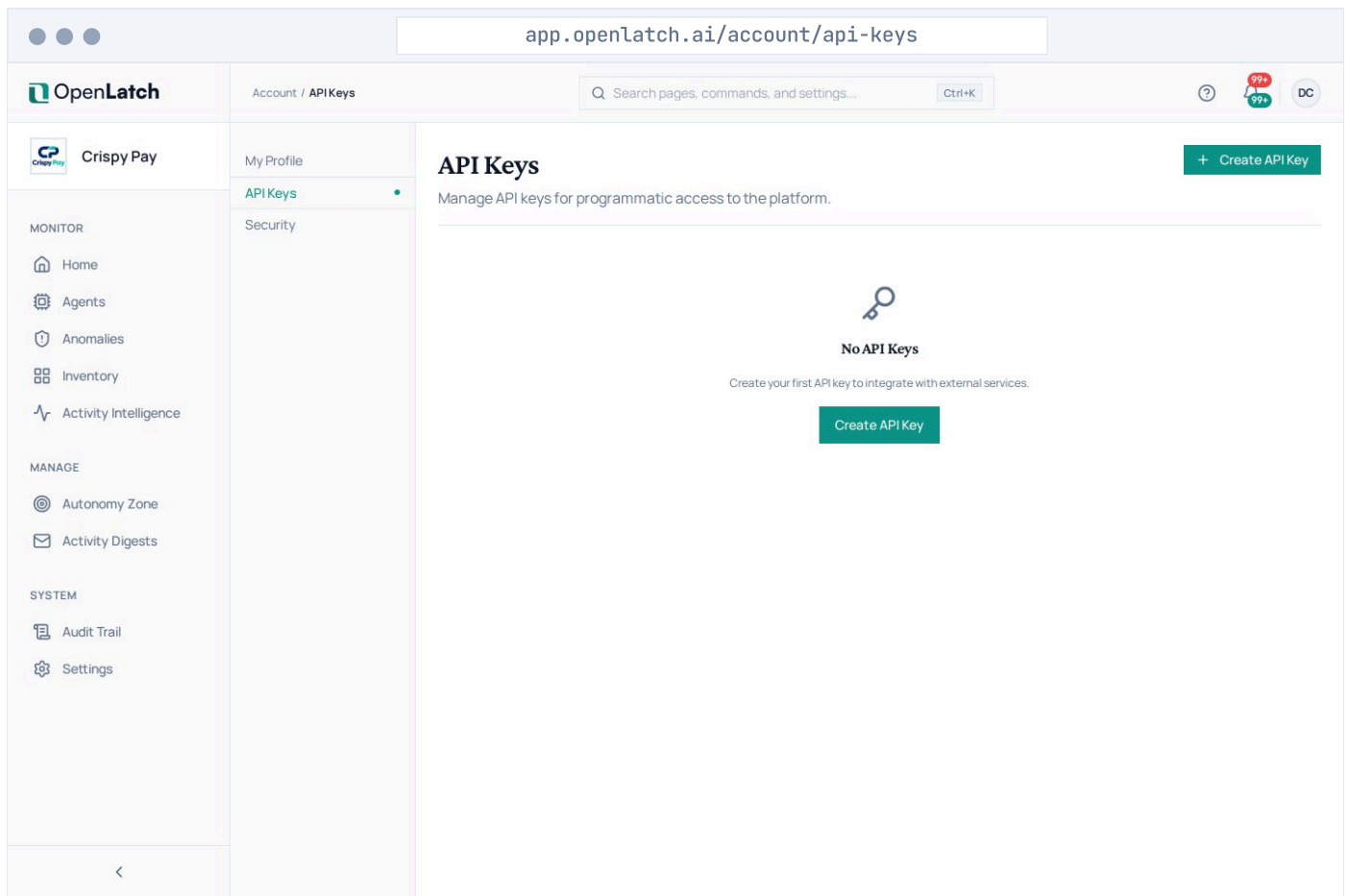
- 2 **Guárdala en el almacén de secretos de tu organización**. La instalación gestionada lee la clave del entorno como `OPENLATCH_API_KEY`. Nunca se escribe en un script, un paquete ni una línea de comandos.

- 3 **Consérvala hasta que las máquinas de la oleada tengan otra**. Cada máquina guarda la clave con la que se inscribió y se autentica con ella, así que revocar una clave de oleada desconecta todas las máquinas de esa oleada. Antes de revocar una, da una clave nueva a sus máquinas: defínela como `OPENLATCH_API_KEY` y vuelve a ejecutar `openlatch init --yes` en cada máquina. La fecha de último uso te indica si una clave se sigue presentando antes de revocarla.

Resultado esperado: la clave figura como revocada, y todo lo que aún la presente, una instalación o una máquina, se rechaza.

Precaución

Una clave en una línea de comandos acaba en el historial del shell, en los listados de procesos y en los registros de la gestión de configuración. El cliente rechaza una credencial pasada así. Proporcióнала por el entorno, desde tu almacén de secretos.



API Keys: cada clave listada con un valor enmascarado tras su creación.

[Gestionar las claves de API](#) describe las claves con más detalle.

En resumen

- OpenLatch crea la organización e invita a su propietario; el asistente de bienvenida es donde le das nombre.
- SAML 2.0 se configura por organización y se comprueba con una prueba que nombra la etapa que falló. Los dominios de correo permitidos y las invitaciones son las dos puertas de la pertenencia.
- Los roles son owner, admin, editor, viewer, y cada persona invitada empieza como viewer. Una conexión de directorio es opcional, de solo lectura, y determina el propietario de un agente.
- Define la conservación del registro de auditoría, con purga automática, antes de empezar el despliegue, y dirige las notificaciones al canal que ya se usa.
- Crea una credencial de cliente por oleada de despliegue, proporcióнала por el almacén de secretos como OPENLATCH_API_KEY y revócala cuando termine la oleada.

Desplegar el cliente

Instala el cliente en cada máquina que ejecuta un agente. El cliente es el punto de decisión en el camino de ejecución del agente: instalarlo conecta cada agente ya presente en la máquina, y desde entonces cada acción cubierta se decide en la máquina, antes de ejecutarse, incluso cuando la consola no es accesible.

Antes de empezar, comprueba estos elementos:

#	Elemento	De dónde viene	Bloquea
1	La organización existe, y las personas que llevan el piloto pueden iniciar sesión	Configurar la organización	Todas las rutas
2	Una clave de API por oleada de despliegue, guardada en tu almacén de secretos	La página API Keys de tu cuenta	Solo las rutas gestionadas
3	Al menos un agente instalado en la máquina de destino y con sesión iniciada en su propia cuenta	El instalador del propio agente	Todas las rutas
4	Tráfico saliente aprobado, con las respuestas de modelos en streaming exentas de almacenamiento en búfer y de inspección	Requisitos de red	Todas las rutas
5	El tipo de proxy vigente en la subred de destino: directo, Basic, Negotiate, PAC, o inspección TLS con su certificado raíz	El equipo que gestiona tu proxy	Las rutas gestionadas

Un agente ausente cuando se ejecuta la instalación no queda conectado. Instala primero los agentes, o vuelve a ejecutar la instalación después; volver a ejecutarla es idempotente, así que repetirla nunca es destructivo.

Elegir una ruta de instalación

Cuatro rutas llevan al mismo estado final. Elige según de dónde viene la credencial y a cuántas máquinas debe llegar la ruta, no según el sistema operativo. La mayoría de los entornos usan dos: la ruta de desarrollador para el piloto y una ruta gestionada para el resto.

Ruta	Ideal para	Alcance	Credencial
Ruta de desarrollador	Oleada 1: un equipo piloto que se inscribe por sí mismo	Unas diez máquinas	Inicio de sesión en el navegador
Script de gestión de equipos, contexto de usuario	Entornos ya gestionados con Intune o Jamf	Todo el parque	Clave de la oleada desde el almacén de secretos
Gestión de configuración o paso de pipeline	Parques Linux y entornos mixtos con automatización existente, como Ansible	Todo el parque	Clave de la oleada desde una consulta a un almacén
Imagen de contenedor	Máquinas de compilación y ejecutores de integración continua	Una inscripción por trabajo	Secreto de ejecución, montado por trabajo

Tres comandos instalan el cliente. Cuál ejecutas depende de lo que ya tiene la máquina, no del tamaño del parque:

Máquina	Comando
macOS o Linux, sin entorno de ejecución	El script <code>install.sh</code> , más abajo
Windows, sin entorno de ejecución	El script <code>install.ps1</code> , más abajo
Cualquier máquina que ya tenga Node.js	<code>npx @openlatch/client init</code> . Node.js es lo que necesita npx; el cliente es un binario nativo y no necesita ningún entorno de ejecución

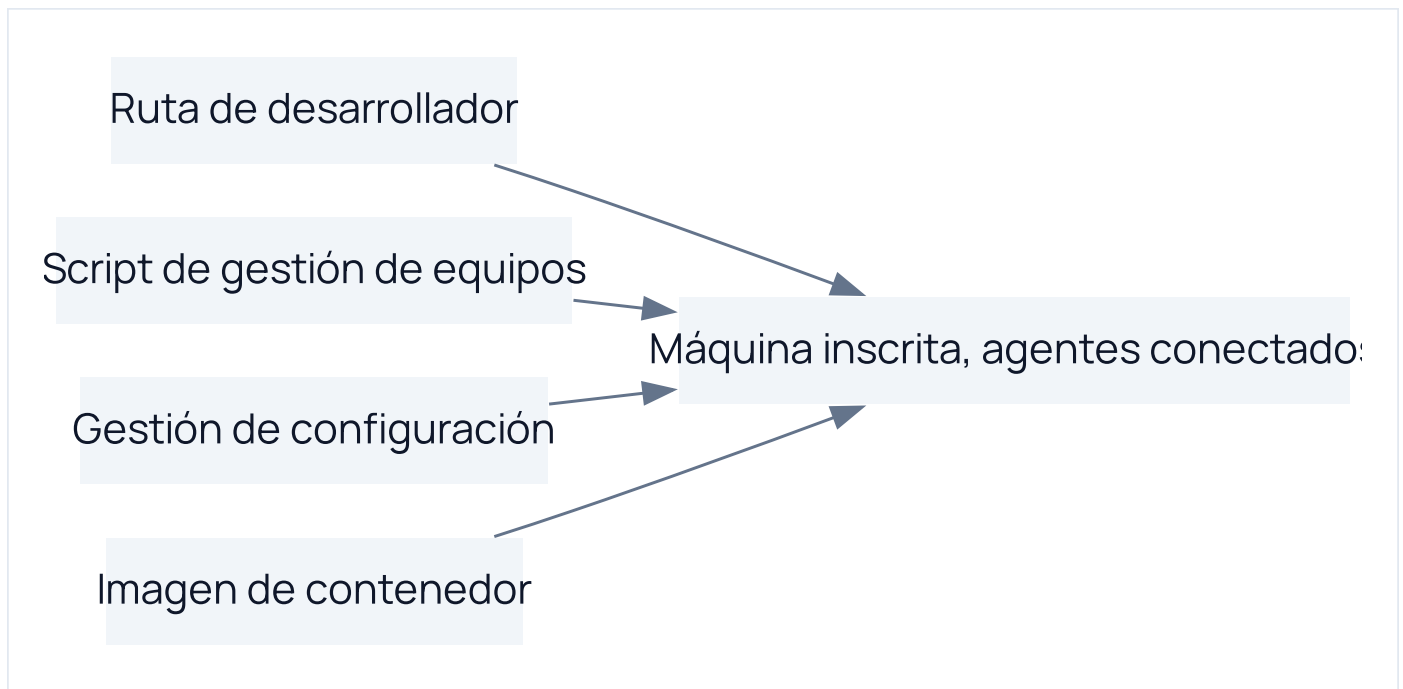


Figura 1. Cuatro rutas de instalación, un único estado final: una máquina inscrita con cada agente conectado.

Instalar una máquina

Un comando inscribe una máquina. Los scripts de instalación llevan el cliente consigo, así que la máquina no necesita ningún entorno de ejecución. Ejecuta la línea de la plataforma que tienes delante.

En macOS y Linux:

```
curl -LsSf https://openlatch.ai/install.sh | sh
```

En Windows:

```
powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
```

En una máquina que ya tiene Node.js:

```
npx @openlatch/client init
```

Los tres llevan al mismo estado final, así que un equipo que ya usa uno lo conserva. Los scripts instalan el cliente en el directorio personal del usuario, añaden ese directorio al PATH y avisan de que el shell actual no lo verá hasta reiniciarse. Después ejecutan `openlatch init` por su ruta completa, para que la inscripción termine en la misma ejecución. Si tu entorno no llega al registro público de paquetes, `OPENLATCH_NPM_REGISTRY` apunta el script y el cliente a tu espejo.

Qué hace la instalación

Paso	Qué ocurre	Resultado esperado
Comprobar la ruta de salida	La comprobación del tráfico saliente va primero: una ruta bloqueada falla antes de cambiar nada en la máquina. Nada queda instalado a medias	Se indica la ruta de salida, con el origen que la definió
Conectar cada agente	Se conecta cada agente instalado en la máquina, según la convención de configuración propia de ese agente. Se conserva el contenido existente de esos archivos	Una línea por agente encontrado, y una línea explícita cuando no se encuentra ninguno
Autenticarse	El orden es <code>OPENLATCH_API_KEY</code> en el entorno, luego una credencial ya presente en el almacén de credenciales del sistema y, por último, el inicio de sesión en el navegador. Un error de red conserva una credencial guardada; solo un rechazo provoca un nuevo inicio de sesión. Con varias organizaciones, el inicio de sesión pregunta cuál, salvo que <code>--org</code> u <code>OPENLATCH_ORG</code> la nombre. Sin nadie a quien preguntar y sin credencial, <code>init</code> nunca inicia una sesión en el navegador y se detiene con <code>0L-1600</code>	Se muestran la cuenta y la organización, o se abre una ventana del navegador en un terminal interactivo
Preguntar por la telemetría de producto	La telemetría está activada por defecto y se puede rechazar; la elección se ofrece antes de escribir ninguna configuración de agente. Contiene nombres de comandos, duraciones e información de errores, nunca contenido	Una sola pregunta, y la elección en el informe final
Arrancar con la sesión del usuario	El cliente se registra en el gestor de servicios del sistema operativo, a nivel de usuario. Un fallo aquí no es fatal: una sesión sin interfaz y un planificador restringido se pueden superar	El informe indica que el cliente volverá a arrancar tras cerrar sesión o reiniciar
Demostrarlo e informar	La instalación confirma que el cliente se está ejecutando, que cada agente conectado responde y que se llega a la consola	Un informe final que nombra todo lo que requiere atención, con la solución al lado

Nota

Algunos agentes piden al desarrollador que confirme la conexión una vez, la primera vez que se ejecutan tras la instalación. Hasta entonces, el agente está conectado en teoría pero no en la práctica. `openlatch status` muestra qué agentes siguen pendientes, y [Niveles de control](#) explica cómo recibe un veredicto cada agente. Una máquina no está terminada hasta que el desarrollador abre el agente una vez.

Confirmar la instalación

```
openlatch status
openlatch doctor
```

Una máquina está instalada cuando `openlatch doctor` no informa de ningún problema y la máquina aparece en el registro de agentes de la consola con sus agentes y un propietario. [Verificar y desplegar](#) cubre esa comprobación máquina a máquina y en toda una oleada.

Detrás de un proxy corporativo

Cada llamada saliente del cliente recorre una única ruta de salida. El tráfico del bucle local de la máquina nunca la recorre. La ruta se resuelve ajuste por ajuste, así que una opción puede fijar la URL del proxy mientras el paquete de certificados viene de otra parte.

Orden	Origen	Úsalo para
1	Las opciones del comando <code>openlatch init</code>	Fijar la ruta en el momento de la inscripción
2	<code>OPENLATCH_PROXY</code> y sus variables hermanas	El único canal autorizado a llevar una credencial de proxy
3	La configuración propia del cliente, escrita en la instalación	Lo que lee el cliente cuando arranca con la sesión, donde no existen variables de shell
4	La familia estándar de variables <code>https_proxy</code>	Un entorno que ya las exporta para todas las herramientas
5	El descubrimiento del sistema operativo	Una inscripción sin configuración en un escritorio gestionado
6	Directo	El último recurso. Se puede rechazar, para que una ruta sin resolver falle en lugar de salir en directo

Lo que permitió la instalación se escribe en la configuración propia del cliente, con su origen. El cliente arranca con la sesión del usuario y no hereda ninguna variable del shell de instalación, así que una ruta que solo existía en una variable de entorno dejaría la máquina saliendo en directo al terminar la sesión.

```
openlatch init --proxy http://proxy.corp:8080 --ca-bundle /etc/ssl/corp-root.pem
openlatch init --proxy http://proxy.corp:8080 --proxy-auth negotiate --proxy-spn
HTTP/proxy.corp
```

El paquete de certificados se combina encima del almacén de confianza del sistema, sin sustituirlo nunca, y la verificación nunca se desactiva. Negotiate usa la identidad de la sesión iniciada, con un nombre de principal de servicio que por defecto es HTTP/ seguido del host del proxy. NTLM se rechaza por diseño, así que un entorno solo NTLM necesita un cambio en el proxy antes de la inscripción.

Se rechaza una credencial pasada dentro de `--proxy`, porque cualquier proceso del host puede leer los argumentos de la línea de comandos. Proporcióнала con `OPENLATCH_PROXY` o responde a la pregunta en un terminal. En ambos casos, la contraseña pasa al almacén de credenciales del sistema, nunca acaba en un archivo en disco y se enmascara en cada línea de registro.

Precaución

En un host detrás de un proxy, las direcciones del bucle local deben estar en la lista de exclusiones del proxy del shell que lanza tus agentes. Algunos agentes reciben esa entrada automáticamente en la instalación. Cuando un agente no tiene dónde recibirla, exporta `NO_PROXY=127.0.0.1,localhost` en el shell que lo lanza. Sin ello, el tráfico del agente hacia el cliente sale hacia el proxy corporativo en lugar de quedarse en la máquina, y el proxy responde con un error.

Instalaciones gestionadas para parques

Precaución

Usa una clave de API por oleada de despliegue, léida de tu almacén de secretos y revocada en la consola cuando termine la oleada. Nunca escribas la clave en un script, un paquete ni una línea de comandos. Cada ejemplo siguiente pone `OPENLATCH_API_KEY` en el entorno de la instalación. Cuando la plataforma de gestión puede pasar un valor a un script, como Jamf con el parámetro 4, usa ese canal; cuando no puede, como Intune, el script obtiene la clave del almacén de secretos como primera acción. Trata el cuerpo de un script de despliegue como legible: una plataforma de gestión lo guarda en texto claro y puede escribirlo en los registros del dispositivo.

Una instalación gestionada ejecuta el mismo script con tres añadidos:

Añadido	Efecto
<code>OPENLATCH_INSTALL_YES</code>	Hace la instalación desatendida
<code>OPENLATCH_NO_INIT</code>	Retiene la inscripción que el script ejecutaría al final, para que tu script ejecute él mismo <code>openlatch init --yes</code> . Con <code>--yes</code> , la inscripción nunca pregunta: falla y nombra lo que necesitaba
<code>OPENLATCH_API_KEY</code> en el entorno	La credencial llega desde tu almacén de secretos en lugar de por un navegador

Ejecuta `openlatch doctor` como última línea de cada script y recoge su salida, para que una máquina que se instaló pero no está totalmente conectada siga visible en lugar de contarse como terminada.

Script de gestión de equipos, contexto de usuario (Windows)

```

# Platform script, user context. Runs silently and cannot prompt.
$ErrorActionPreference = "Stop"

# The platform has no field that sets an environment variable, so the script
# fetches the wave key itself, as its first action. Replace this call with the
# one your secret store provides.
$env:OPENLATCH_API_KEY = Get-WaveKeyFromSecretStore
if (-not $env:OPENLATCH_API_KEY) {
    Write-Output "the wave key could not be read from the secret store"; exit 1
}

$env:OPENLATCH_INSTALL_YES = "1"
$env:OPENLATCH_NO_INIT     = "1"

powershell -ExecutionPolicy Bypass -c "irm https://openlatch.ai/install.ps1 | iex"
if ($LASTEXITCODE -ne 0) { exit $LASTEXITCODE }

$openlatch = "$env:LOCALAPPDATA\openlatch\bin\openlatch.exe"
& $openlatch init --yes
if ($LASTEXITCODE -ne 0) { exit $LASTEXITCODE }
& $openlatch doctor
exit $LASTEXITCODE

```

Intune ejecuta los scripts en silencio y no admite instalaciones interactivas: por eso la instalación es desatendida y --yes es obligatorio. La extensión de gestión obtiene las asignaciones según su propio calendario, así que planifica una oleada en días más que en minutos. Un script en contexto de usuario necesita un usuario con sesión iniciada, y los scripts de plataforma no llegan a los dispositivos que solo están registrados en Microsoft Entra ID; para llegar a los dispositivos personales registrados, usa la ruta de aplicación Win32 con un comportamiento de instalación User, que el diseño sin elevación hace posible.

Ningún ajuste de Intune define una variable de entorno para un script de plataforma. Por eso la llamada al almacén de secretos es la primera acción del script. Pasar la clave como argumento no es una opción: el cliente rechaza una credencial en sus propios argumentos.

Script de gestión de equipos, contexto de usuario (macOS)

Jamf pasa su primer valor personalizado en el parámetro 4, lo que mantiene la clave fuera del cuerpo del script. La política no necesita ningún paso de empaquetado propio, porque la instalación se registra en el

directorio personal del usuario. Asocia la política a un desencadenante a nivel de usuario para que la inscripción se ejecute como el desarrollador y no como root.

```
#!/bin/zsh
set -eu

# Parameter 4 carries the wave key, or replace this line with a secret-store call.
export OPENLATCH_API_KEY="${4:?wave key was not supplied}"

export OPENLATCH_INSTALL_YES=1
export OPENLATCH_NO_INIT=1

curl -LsSf https://openlatch.ai/install.sh | sh

"$HOME/.local/bin/openlatch" init --yes
"$HOME/.local/bin/openlatch" doctor
```

Gestión de configuración o paso de pipeline

```
- name: Install the OpenLatch client
  ansible.builtin.shell: curl -LsSf https://openlatch.ai/install.sh | sh
  args:
    creates: "{{ ansible_env.HOME }}/.local/bin/openlatch"
  environment:
    OPENLATCH_INSTALL_YES: "1"
    OPENLATCH_NO_INIT: "1"

- name: Enroll the machine
  ansible.builtin.command: "{{ ansible_env.HOME }}/.local/bin/openlatch init --yes"
  environment:
    # Or a vault lookup, so the value never lands in the play or in a fact.
    OPENLATCH_API_KEY: "{{ lookup('env', 'OPENLATCH_API_KEY') }}"

- name: Report what the machine looks like
  ansible.builtin.command: "{{ ansible_env.HOME }}/.local/bin/openlatch doctor"
  changed_when: false
```

Ejecuta la inscripción en una muestra de la oleada antes de la ejecución completa, y lee la salida de la tarea de informe en esas máquinas. Un tipo de proxy que la instalación no sabe descubrir aparece ahí, y no en todas las máquinas a la vez.

Imagen de contenedor e integración continua

```
FROM debian:bookworm-slim
RUN apt-get update \
    && apt-get install -y --no-install-recommends ca-certificates curl \
    && rm -rf /var/lib/apt/lists/*

ENV OPENLATCH_INSTALL_YES=1 \
    OPENLATCH_NO_INIT=1 \
    OPENLATCH_NO_MODIFY_PATH=1 \
    OPENLATCH_INSTALL_DIR=/usr/local/bin

RUN curl -LsSf https://openlatch.ai/install.sh | sh
COPY entrypoint.sh /usr/local/bin/entrypoint.sh
ENTRYPOINT ["/usr/local/bin/entrypoint.sh"]
```

entrypoint.sh

```
#!/bin/sh
set -eu
: "${OPENLATCH_API_KEY:?the runtime secret was not mounted}"
openlatch init --yes --no-persistence
exec "$@"
```

```
- name: Enroll the runner
  env:
    OPENLATCH_API_KEY: ${ secrets.OPENLATCH_API_KEY }
    OPENLATCH_INSTALL_YES: "1"
    OPENLATCH_NO_INIT: "1"
  run: |
    curl -LsSf https://openlatch.ai/install.sh | sh
    "$HOME/.local/bin/openlatch" init --yes --no-persistence
    "$HOME/.local/bin/openlatch" status
```

La imagen base no lleva nada para el cliente: el script trae el cliente consigo. La credencial se monta en tiempo de ejecución, así que ninguna capa de la imagen la contiene. `OPENLATCH_NO_MODIFY_PATH` impide que la instalación escriba una línea `PATH` en un perfil de shell que ningún shell interactivo leerá, y `OPENLATCH_INSTALL_DIR` coloca el cliente donde el `PATH` de la imagen ya busca. `--no-persistence` omite el gestor de servicios, porque un contenedor no tiene sesión de usuario en la que arrancar. La telemetría de

producto está desactivada en integración continua sin configuración. Los ejecutores efímeros se inscriben en cada trabajo, y como repetir la instalación es seguro, reintentar un trabajo también lo es.

Qué instala el cliente en una máquina

El cliente se instala y se ejecuta a nivel de usuario. No se necesitan derechos de administrador, no se escribe nada en un directorio del sistema y no se registra ningún servicio a nivel de máquina.

Elemento	Detalle
Una única unidad instalada	El cliente de OpenLatch, instalado una vez por máquina que ejecuta un agente
Dónde se coloca	Los scripts colocan el cliente en <code>~/ .local/bin</code> en macOS y Linux, y en <code>%LOCALAPPDATA%\openlatch\bin</code> en Windows, y añaden ese directorio al PATH. <code>OPENLATCH_INSTALL_DIR</code> lo cambia de sitio. A propósito, no es el directorio de estado de la fila siguiente
Archivos	El cliente guarda su configuración y sus registros locales en un único directorio: <code>~/ .openlatch</code> en macOS y Linux, <code>%APPDATA%\openlatch</code> en Windows. <code>OPENLATCH_DIR</code> lo cambia de sitio. Todo lo sensible que contiene solo lo puede leer su propietario
Configuración de los agentes	El archivo de configuración de cada agente conectado recibe una entrada que apunta al cliente. Se conserva el contenido existente, y lo que había antes se registra para poder restaurarlo
Arranque	El cliente arranca con la sesión del usuario, bajo el gestor de servicios del sistema operativo
Credenciales	La clave de API, y cualquier credencial de proxy, se guardan en el almacén de credenciales del sistema
Postura de red	Solo escucha en la interfaz de bucle local. No se abre ningún puerto entrante

`openlatch uninstall` desconecta cada agente que encuentra en la máquina, detiene el cliente y restaura lo que cambió en la configuración de cada agente. `openlatch uninstall --purge` elimina además el directorio de estado y la credencial guardada. Una purga no elimina los archivos instalados: están fuera del

directorio de estado, así que una purga los deja, junto con la línea PATH que añadió el script. [Operar](#) cubre la eliminación completa.

En resumen

- Cuatro rutas de instalación llevan a un único estado final. Elige según de dónde viene la credencial y a cuántas máquinas debe llegar la ruta, no según el sistema operativo.
- Los scripts de instalación inscriben una máquina en macOS, Windows y Linux sin ningún entorno de ejecución; `npx @openlatch/client init` llega al mismo estado final donde Node.js está presente. En todos los casos, la instalación detecta los agentes presentes y conecta cada uno, y volver a ejecutarla es seguro.
- Algunos agentes piden al desarrollador que confirme la conexión una vez. Una máquina no está terminada hasta que eso ocurre.
- Una máquina está instalada cuando `openlatch doctor` no informa de ningún problema y la máquina aparece en el registro de agentes de la consola con sus agentes y un propietario.
- Una clave de API por oleada, leída de tu almacén de secretos en tiempo de ejecución, nunca en un script, un paquete ni una línea de comandos.

Verificar y desplegar

Una máquina está desplegada cuando `openlatch doctor` no informa de ningún problema y la consola tiene su registro. Esta página demuestra ambas cosas y después amplía el despliegue en tres oleadas.

Verificar una máquina

Ejecuta los dos comandos en cada máquina que instales, en este orden.

```
openlatch status
openlatch doctor
```

`openlatch status` responde a una pregunta: ¿está esta máquina cubierta ahora mismo? Un buen resultado muestra la máquina conectada a la consola, políticas presentes en la máquina y cada agente de la máquina conectado.

`openlatch doctor` responde a la pregunta más difícil: ¿es cierto ese informe? Muestra una causa y una solución para cada cosa que encuentra. Un buen resultado es que no encuentre ningún problema. Es el paso de aceptación de una máquina, así que cuenta una máquina con un problema sin resolver como una máquina fallida en el informe de la oleada. Cuando informe de algo, sigue la solución que indica y vuelve a ejecutarlo, hasta que salga limpio.

Qué confirmas	Cómo es un buen resultado
La máquina está conectada	Ha llegado a la consola y tiene allí un registro actualizado
Las políticas están presentes	Hay políticas en la máquina. Es lo que le permite seguir decidiendo cuando la consola no es accesible
Los agentes están conectados	Cada agente instalado en la máquina está conectado, y ninguno sigue pendiente de que su desarrollador confirme la conexión

Precaución

Una máquina puede estar en marcha sin cubrir nada, y desde fuera parece idéntica a una que funciona: el cliente está en marcha, la consola tiene un registro y uno de los agentes de la máquina nunca se conectó. No pases al siguiente paso del despliegue tras una instalación que `openlatch doctor` no haya validado.

Crear un archivo de diagnóstico

Cuando una máquina tiene un problema que no puedes resolver con la solución que muestra, crea un archivo de diagnóstico.

```
openlatch doctor --rescue
```

El comando genera un único archivo ZIP depurado con el diagnóstico, la configuración del cliente y una ventana reciente de los registros locales de la máquina.

Opción	Qué hace
<code>--since 24h</code>	Limita la ventana de registros locales que incluye el archivo
<code>--yes</code>	Omite la confirmación, para una ejecución desatendida

El archivo permite investigar un problema a partir de las pruebas de la propia máquina y no de una descripción del síntoma. Está depurado con ese fin: envía el archivo y no el directorio en sí.

Nota

Abre un caso de soporte con el archivo adjunto, el sistema operativo de la máquina y la hora del problema con su zona horaria. Contacta con tu responsable técnico de OpenLatch por el canal acordado con tu organización. Envía los avisos de seguridad a security@openlatch.ai.

Confirmar en la consola

Un informe limpio en la máquina es la mitad de la comprobación. La otra mitad es que la consola tenga un registro de la máquina, con los agentes que detectó y la persona a la que se atribuyen sus acciones.

- 1 **Abre el registro de agentes.** Inicia sesión en `app.openlatch.ai` y abre **Agents**.

Resultado esperado: una fila por cada agente que ha enviado datos.

- 2 **Busca la máquina.** Busca por nombre de host o por el usuario del sistema operativo con el que se ejecutó la instalación.

Resultado esperado: la máquina aparece en las 24 horas siguientes a la instalación, y en minutos en una red sana.

- 3 **Confirma los agentes de la máquina.** Abre un agente y lee sus detalles, incluido su nivel de control.

Resultado esperado: cada agente instalado en esa máquina aparece, cada uno con su nivel de control. Un agente que falta es un problema de instalación, no de la consola: vuelve a las comprobaciones de la máquina de arriba.

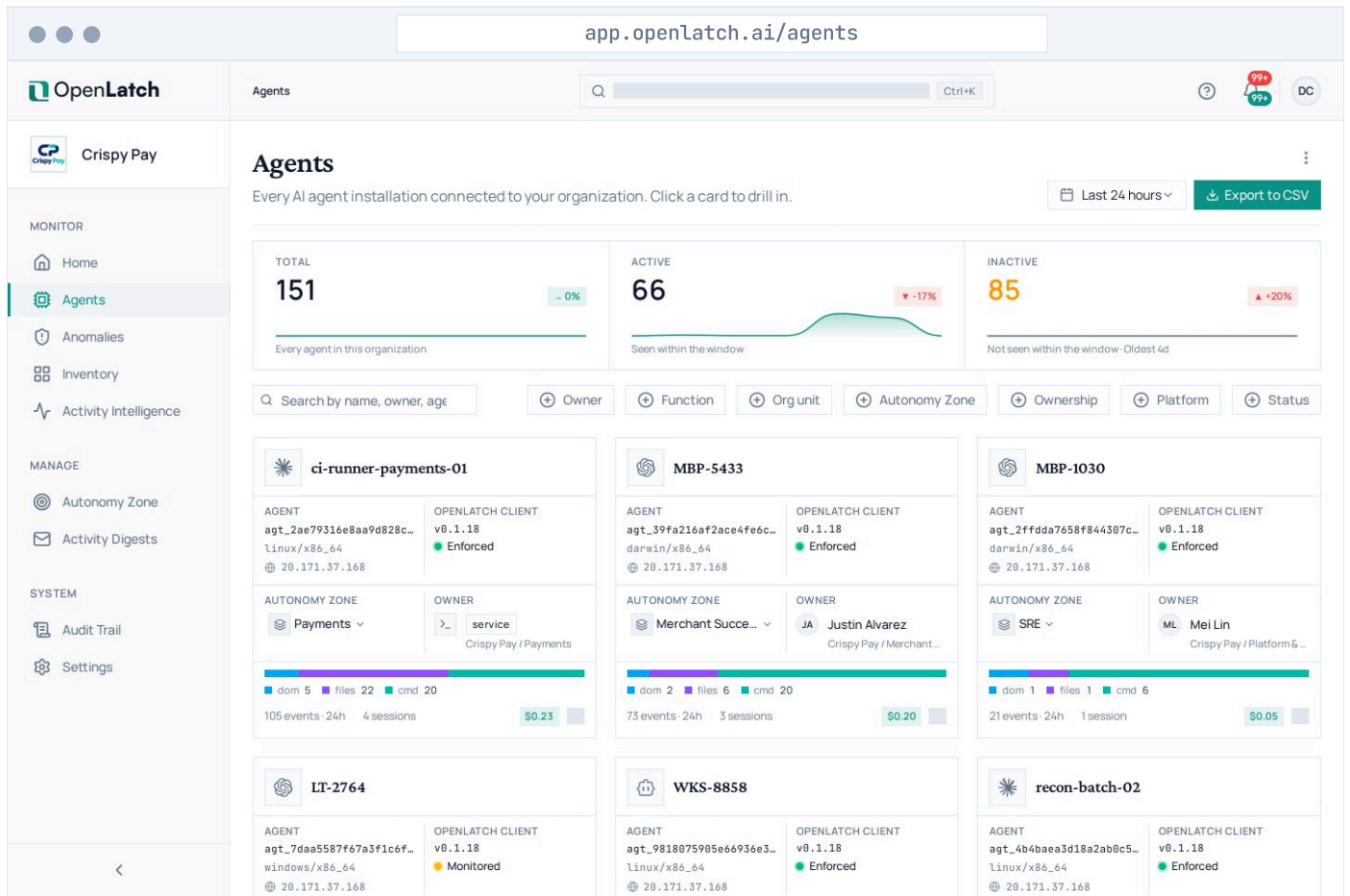
- 4 **Confirma el propietario determinado.** La pestaña **Identity** del agente nombra a la persona a la que se atribuyen sus acciones, con su cargo y su lugar en la organización. Con un directorio conectado, el propietario se determina mediante una búsqueda de solo lectura; sin él, OpenLatch se basa en lo que envía la máquina.

Junto al nombre hay dos controles: uno explica cómo se llegó a ese propietario y el otro permite sustituirlo a mano. Corregir la propiedad es tarea del rol editor.

Resultado esperado: una persona con nombre en lugar de un campo vacío.

- 5 **Confirma que llegan los registros.** Lee la hora de última actividad del agente mientras un desarrollador trabaja en esa máquina.

Resultado esperado: la hora avanza. Un registro que deja de avanzar apunta a la ruta de salida de la máquina.



Agents: una fila por cada agente que ha enviado datos, con su nivel de control y su propietario.

The screenshot displays the OpenLATCH web application interface. The top navigation bar shows the URL `app.openlatch.ai/agents/ci-runner-payments-01/iden...`. The left sidebar contains a menu with sections: MONITOR (Home, Agents, Anomalies, Inventory, Activity Intelligence), MANAGE (Autonomy Zone, Activity Digests), and SYSTEM (Audit Trail, Settings). The 'Agents' section is highlighted. The main content area is titled 'Agents / Agent / Identity' and shows the 'Identity' tab for the agent 'ci-runner-payments-01'. The agent's status is 'IDLE'. The 'Identity' section describes the human accountable for the agent and shows the ownership hierarchy: Jenkins CI (service) owns Camille Lefèvre (person), who owns the agent. The ownership is resolved by the user.

La pestaña Identity de un agente: la persona a la que se atribuyen sus acciones y su lugar en la organización.

Desplegar por oleadas

Tres oleadas de dos semanas, cada una demostrando algo de lo que depende la siguiente. La oleada 1 demuestra la instalación por la ruta de desarrollador. La oleada 2 demuestra la ruta gestionada. La oleada 3 demuestra el perímetro. Emite una credencial de cliente por oleada y revócala en la consola cuando termine la oleada.

Atributo	Oleada 1	Oleada 2	Oleada 3
Alcance	Unas diez máquinas de desarrolladores	Unas cincuenta máquinas	El perímetro completo
Ruta de instalación	Ruta de desarrollador, en el terminal del propio desarrollador	Ruta gestionada, desatendida	Ruta gestionada, desatendida
Duración	Dos semanas	Dos semanas	Dos semanas
Actividad	Inscribir cada máquina, verificarla, perseguir los agentes que siguen pendientes de su desarrollador y anotar los problemas encontrados	Empaquetar la instalación, demostrarla en una máquina de cada sistema operativo, ejecutarla desatendida y leer el informe que recoge	Ejecutar el mismo paquete en todo el perímetro y perseguir las máquinas que nunca envían datos
Criterios de entrada	La organización está configurada y se ha emitido una credencial para la oleada	La oleada 1 cumplió sus criterios de salida, y el script gestionado se ejecuta limpio en una máquina de cada sistema operativo	La ruta gestionada está demostrada en cada sistema operativo del parque, y el tipo de proxy está confirmado en cada subred del alcance
Criterios de salida	Cada máquina inscrita envía datos a la consola en las 24 horas siguientes a la instalación; <code>openlatch doctor</code> no informa de ningún problema en ninguna; ningún agente queda pendiente de su desarrollador; la ruta del proxy está confirmada en cada sistema operativo de la oleada	Los mismos cuatro, más cada máquina instalada sin que un desarrollador abra un terminal	Los mismos cuatro, más la decisión de pasar a operación habitual
Marcha atrás	<code>openlatch uninstall</code> en la máquina afectada	Lo mismo, entregado por la herramienta que lo instaló	Lo mismo, entregado por la herramienta que lo instaló



Figura 1. Tres oleadas, lo que demuestra cada una y la puerta entre ellas.

Problemas más probables

Síntoma	Causa probable	Qué hacer
La máquina envía datos, pero uno de sus agentes no está conectado	Ese agente no estaba instalado cuando se ejecutó la instalación, o su configuración está en una ruta personalizada	Instala el agente y vuelve a ejecutar <code>openlatch init</code> en la máquina. El cliente ya está allí, así que no hace falta descargar nada. Si la ruta es personalizada, defínela en el entorno de la inscripción
Un agente figura como pendiente	Algunos agentes piden al desarrollador que confirme la conexión una vez, la primera vez que se ejecutan tras la instalación	Pide al desarrollador que abra el agente una vez y confirme. Si el entorno bloquea la configuración de los agentes de forma centralizada, entrega la conexión por ese canal gestionado
Las políticas nunca llegan a la máquina	La máquina no ha llegado a la consola desde que se instaló, o su credencial se revocó o se renovó	Confirma primero la ruta de salida, según Requisitos de red . Si la credencial se rechazó, emite una para la oleada y vuelve a ejecutar la instalación
La máquina está conectada, pero sus agentes se comportan como si no hubiera nada	Los puntos de acceso de modelos de los agentes están bloqueados en el proxy, o sus respuestas en streaming se almacenan en búfer	Exime esos puntos de acceso del almacenamiento en búfer de respuestas y de la inspección de flujos. Son el tráfico propio de los agentes y deben seguir siendo accesibles como antes
La máquina nunca aparece en la consola	La ruta de salida está bloqueada, o la inscripción no llegó a completarse	Ejecuta <code>openlatch doctor</code> en la máquina. Nombra lo que falló, lo que distingue un problema de proxy de una instalación que se detuvo antes de tiempo

En resumen

- Una instalación está verificada cuando `openlatch doctor` no informa de ningún problema y la consola muestra los agentes de la máquina, cada uno con un nivel de control y un propietario determinado.
- `openlatch status` indica si la máquina está cubierta. `openlatch doctor` lo demuestra, con una causa y una solución para cada problema detectado.
- `openlatch doctor --rescue` crea un archivo depurado con el diagnóstico, la configuración y una ventana reciente de los registros locales. Abre cada caso de soporte con él.
- Un agente que sigue pendiente de que su desarrollador confirme la conexión está conectado en teoría pero no en la práctica. Persigue esos casos antes de cerrar una oleada.
- Tres oleadas de dos semanas: unas diez máquinas de desarrolladores, unas cincuenta por la ruta gestionada y, después, el perímetro. Una credencial de cliente por oleada, revocada cuando termina la oleada.

Operar

Usa esta página una vez inscritas las máquinas: para mantener el cliente actualizado, trabajar con los registros que guarda en cada máquina y desinstalarlo.

Actualizaciones

El cliente puede comprobar si hay una versión nueva y actualizarse. La descarga se comprueba en integridad antes de sustituir nada en la máquina: se compara un resumen SHA-512 con el que publica el registro de paquetes para esa versión, así que nunca se aplica una descarga dañada o truncada. Esa comprobación detecta daños y truncamientos; no protege frente a un registro comprometido. Deja activada la actualización automática en las instalaciones que hace un desarrollador por sí mismo, donde mantiene la máquina al día sin que nadie la programe.

Un entorno gestionado suele querer controlar la versión. Desactiva la actualización automática y deja activada la comprobación, para que una máquina siga indicando que hay una versión más reciente mientras espera tu ventana de mantenimiento. Después entrega la nueva versión como se entregó el cliente la primera vez, con las herramientas que ya gestionan la máquina.

Variable	Valor
OPENLATCH_AUTO_UPDATE	<code>false</code> se impide que el cliente se actualice solo. Defínela donde quieras fijar la versión
OPENLATCH_UPDATE_CHECK	Déjala en <code>true</code> , para que <code>openlatch status</code> siga indicando que hay una versión más reciente

Nota

Sustituir los archivos en disco no hace por sí solo que el cliente en ejecución pase a la nueva versión. La máquina indica la diferencia hasta que el cliente se reinicia, así que termina la ventana con `openlatch restart`, o con el cierre e inicio de sesión que tu entorno ya programa.

Registros locales

El cliente guarda su configuración y sus registros locales en un único directorio: `~/ .openlatch` en macOS y Linux, `%APPDATA%\openlatch` en Windows. `OPENLATCH_DIR` lo cambia de sitio. Un entorno que lo cambia de sitio debe apuntar sus propios scripts de recogida y eliminación al mismo valor.

Los registros se escriben con las credenciales enmascaradas, y las rutas de archivo se enmascaran de la misma forma. [Tratamiento de datos y seguridad](#) enumera lo que se captura, lo que se enmascara y cuánto tiempo se conserva cada tipo. Los registros que aún no han llegado a la consola se guardan en ese directorio y se envían cuando la consola vuelve a ser accesible. Esa reserva está acotada: por encima de su límite de tamaño, 100 MB por defecto, se descartan los registros no enviados más antiguos y el registro del daemon anota un aviso 0L-1204. Consulta los ajustes del outbox y del registro de respaldo en [Configuración](#).

Abrir un caso de soporte

Ejecuta primero `openlatch doctor` en la máquina afectada. Describe lo que encuentra en términos sencillos. Después crea el archivo depurado con `openlatch doctor --rescue`: contiene el diagnóstico, la configuración y una ventana reciente de los registros locales. Envía ese archivo, nunca el directorio en sí, con el sistema operativo y la hora del problema con su zona horaria.

Contacta con tu responsable técnico de OpenLatch por el canal acordado con tu organización. Envía los avisos de seguridad a security@openlatch.ai.

Desinstalar el cliente

Para retirar el cliente de una máquina:



```
openlatch uninstall
```

Para retirar el cliente y los registros que guardó localmente:



```
openlatch uninstall --purge
```

`openlatch uninstall` desconecta el cliente de cada agente que encuentra en la máquina y lo detiene. Cada agente queda como estaba antes de la instalación, y los ajustes que OpenLatch no escribió no se tocan. La configuración y los registros locales se quedan en disco salvo que pidas eliminarlos.

--purge elimina además ese directorio y la credencial guardada en el almacén de credenciales del sistema. Rechaza cualquier directorio que no sea el del cliente, y resuelve los enlaces simbólicos antes de eliminar, para que un enlace no pueda desviar la eliminación a otro sitio. Un entorno que cambia el directorio de sitio debe mantener la misma comprobación en su propio script de eliminación.

Ninguno de los dos comandos elimina los archivos del propio cliente. Se instalaron fuera del directorio de estado, así que una purga los deja en la máquina, junto con la línea PATH que añadió la instalación. Para retirar el cliente por completo, elimina los dos archivos que colocó la instalación, `openlatch` y `openlatch-hook` en `~/ .local/bin` en macOS y Linux, o `openlatch.exe` y `openlatch-hook.exe` en `%LOCALAPPDATA%\openlatch\bin` en Windows. Deja el directorio y todo lo demás que contenga: otro software puede usarlo. En Windows, el directorio pertenece al cliente, así que también puedes eliminarlo junto con la entrada PATH que añadió la instalación; en macOS y Linux, conserva la línea PATH si otras herramientas dependen de `~/ .local/bin`. No hay script de desinstalación: un entorno que quiera que se haga por él lo hace con las herramientas que instalaron el cliente.

La eliminación es máquina a máquina. No hay que deshacer nada en la consola para una máquina desinstalada, y una máquina que vuelve se instala igual que la primera vez.

En resumen

- Deja activada la actualización automática en las instalaciones de los desarrolladores. Fíjala con `OPENLATCH_AUTO_UPDATE=false` en un entorno gestionado, y deja activada la comprobación para que una máquina siga indicando lo que le falta.
- Una versión nueva entra en vigor cuando el cliente se reinicia: termina la ventana de mantenimiento con `openlatch restart` o con un cierre e inicio de sesión.
- La configuración y los registros locales están en un único directorio. Un caso de soporte se abre con el archivo depurado que genera `openlatch doctor --rescue`.
- `openlatch uninstall` deja cada agente como estaba. --purge elimina además el directorio y la credencial guardada, y deja los dos archivos instalados, que una eliminación completa quita por separado.

Tratamiento de datos y seguridad

Entrega esta página a una revisión de seguridad. Expone qué sale de una máquina que ejecuta el cliente, qué se queda en ella, cuánto tiempo se conserva cada registro y cómo se controla el acceso a la consola.

OpenLatch controla un agente que se comporta mal: uno que entra en bucle o gasta presupuesto en un trabajo que nadie pidió, uno al que se le ha inyectado algo a través de su contexto y actúa en consecuencia con sus herramientas, y uno que incumple una política aunque por lo demás se comporte como se diseñó. Esos fallos causan incidentes reales, y son lo que el bucle de control intercepta antes de que la acción se ejecute. OpenLatch no afirma resistir a un agente diseñado para evadirlo.

Principios

Principio	Qué significa
La decisión se toma en la máquina	Cada veredicto se calcula en la máquina, antes de que la acción se ejecute. La consola no está en el camino de decisión
OpenLatch no está en ningún camino de red	El cliente no es una pasarela y no inspecciona el tráfico de tu organización. Tus agentes llegan a sus puntos de acceso de modelos exactamente como antes
Datos mínimos	Los valores se enmascaran y las cadenas se truncan antes de que nada salga de la máquina, y se envían metadatos siempre que el contenido no sea necesario
Ninguna credencial sale de la máquina	Las claves que usan tus agentes se quedan en la máquina. Las contraseñas de proxy las guarda el sistema operativo, nunca en un archivo de configuración ni en un registro

Inventario de datos

Lee primero la segunda columna.

Dato	Sale de la máquina	Enmascaramiento y límites	Dónde se guarda	Conservación
Metadatos de la acción (nombre de herramienta, sesión, veredicto, momento)	Sí	Ninguno. Los metadatos no contienen texto libre	Consola	180 días
Comandos de shell y argumentos de herramientas	Sí	Credenciales enmascaradas con marcadores etiquetados. Cada cadena limitada a 4 KB	Consola, y en forma de hash en los agregados de uso	180 días; agregados 30 días
Rutas de archivo	Sí	Enmascaradas de la misma forma, y con hash con una clave propia de la organización en los agregados de uso	Consola, y en forma de hash en los agregados de uso	180 días; agregados 30 días
Contenido de archivos	Configurable	Filtrado por defecto, o solo hash. Captura en línea limitada a 64 KB	Consola	180 días
Prompts	Solo metadatos	Aparición, momento, sesión y longitud. El texto del prompt no se envía salvo que tu organización lo active	Consola	180 días
Peticiones y respuestas de modelos	No	El contenido nunca se envía. Se registran el número de tokens, el modelo usado y el reparto de caché	Consola	90 días
Identidad del usuario	Sí	Usuario del sistema operativo y correo de git,	Consola	180 días, con el registro

Dato	Sale de la máquina	Enmascaramiento y límites	Dónde se guarda	Conservación
		para atribuir una acción a una persona		
Identidad de la máquina	Sí	Sistema operativo, arquitectura, direcciones IP local y pública, versiones de software, un identificador de instalación	Consola	180 días, con el registro
Identificador de cuenta de modelo	Sí	Un identificador de cuenta, o una marca que indica que se usa una clave compartida. Nunca un token ni una clave	Consola	180 días, con el registro
Políticas	Solo descendente	Se envían a la máquina, nunca se suben desde ella	En la máquina	Hasta que un conjunto más reciente las sustituye
Veredictos y anomalías	Sí	Ninguno	Consola	180 días

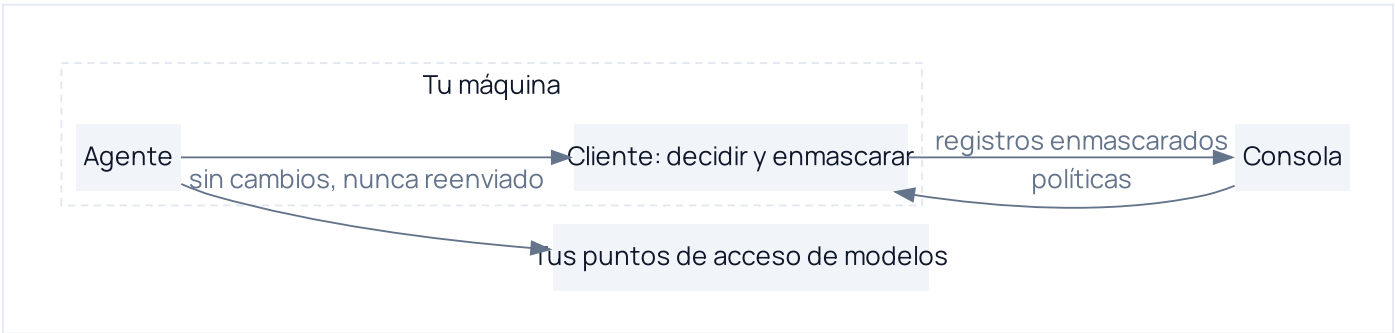


Figura 1. Flujo de datos y límites de confianza: los registros enmascarados suben, las políticas bajan y el tráfico de modelos nunca pasa por OpenLatch.

Enmascaramiento y lo que se queda en la máquina

El enmascaramiento se ejecuta en el cliente, en la máquina, antes de que un registro entre en la cola de envío. Recorre el contenido y sustituye los valores con forma de credencial por un marcador etiquetado que conserva el prefijo público reconocible y elimina el secreto, con la forma [AWS_KEY:AKIA***]. Solo se enmascaran los valores de cadena; los nombres de campo nunca. El enmascaramiento se ejecuta antes del truncamiento, así que cortar una cadena en el límite de 4 KB no puede revelar el final de un secreto parcialmente enmascarado. Se ejecuta después del veredicto, así que el texto depurado nunca cambia qué políticas coinciden. Tu organización puede añadir sus propias formas de credencial a la configuración del cliente, y se aplican junto con las integradas.

 **Precaución**

El cliente guarda en texto claro, en la máquina, los registros locales de las acciones, en ~/ .openlatch en macOS y Linux o %APPDATA%\openlatch en Windows. Contienen el texto de los comandos y las rutas de archivo enmascarados, y los protegen los permisos de archivo del usuario y el cifrado de disco que aplique tu entorno. Las credenciales son la excepción: las guarda el almacén de credenciales del sistema operativo cuando existe y, si no, un archivo que solo puede leer su propietario.

Conservación

Dato	Periodo	Configurable	Exportación
Registros de acciones	180 días	No	—
Registros de decisiones y anomalías	180 días	No	CSV
Agregados de uso	30 días	No	Consola
Archivos con hash	90 días	No	Consola
Registros de coste y tokens	90 días	No	Consola
Registro de auditoría	90, 180, 365 o 730 días, o un periodo personalizado	Sí, con purga automática	CSV o JSON

El registro de auditoría es el único periodo que defines tú, en la consola, con purga automática; [Configurar la organización](#) muestra dónde. Todos los demás periodos son fijos. Las políticas guardadas en la máquina no

tienen periodo de conservación: se sustituyen cuando el cliente obtiene un conjunto más reciente y se eliminan cuando el cliente se desinstala con `--purge`.

Alojamiento

La consola es un servicio alojado en Estados Unidos, en la región US-East. Los datos de cada organización están separados de los de todas las demás por la propia base de datos: la seguridad a nivel de fila se impone en cada tabla de cliente, sin ningún rol de ejecución que la eluda. La lista de subencargados está disponible bajo petición.

En tránsito, el cliente y la consola se comunican con TLS validado frente al almacén de confianza del sistema operativo, con la opción de confiar en una autoridad de certificación adicional donde tu entorno termina TLS con la suya, y sin ninguna opción que desactive la verificación. En reposo, los secretos almacenados están cifrados.

Identidad y acceso

La consola admite el inicio de sesión único SAML 2.0 por organización. Proporcionas el identificador de entidad del proveedor de identidad, su URL de inicio de sesión y un certificado de firma o una URL de metadatos, y después defines los dominios de correo permitidos. Una prueba realiza el intercambio e informa del resultado antes de que la conexión entre en servicio. Microsoft Entra ID, Okta, PingOne y Active Directory Federation Services se conectan así.

Dos puertas controlan quién se une. Los dominios de correo permitidos restringen qué direcciones acepta la consola, y las invitaciones controlan quién es admitido. Una conexión SAML propia de la organización es la única fuente de identidad que admite a un miembro sin invitación, así que un inicio de sesión social nunca sirve para eludir el aprovisionamiento.

Rol	Qué permite
owner	Todo lo que permite admin, más la propiedad de la propia organización
admin	Ajustes de identidad, políticas, credenciales del cliente y ajustes de conservación
editor	Políticas y sustituciones del propietario de un agente
viewer	Acceso de lectura a agentes, ejecuciones, decisiones y el registro de auditoría. Las personas invitadas empiezan aquí

Una credencial de cliente hereda el rol de la persona que la emitió. Una emitida por un editor no puede cambiar un ajuste de identidad, y revocar la credencial o rebajar el rol de la persona cierran la misma puerta.

El registro de auditoría es de solo anexo y está encadenado por hash, así que una fila eliminada o modificada rompe la cadena. Cada cambio de política escribe una fila que nombra al autor, con el estado antes y después. El registro se exporta en CSV o JSON, su periodo de conservación es configurable con purga automática, y la propia exportación escribe un evento de auditoría.

Proteger el cliente y sus políticas

Las actualizaciones del cliente se comprueban en integridad antes de sustituir nada: el resumen SHA-512 de la descarga debe coincidir con el que publica el registro de paquetes para esa versión. Esa comprobación detecta daños y truncamientos; no protege frente a un registro comprometido.

Las políticas guardadas en la máquina se comprueban antes de usarse, y de nuevo en cada carga, no solo cuando se obtienen. Un conjunto que no supera la comprobación se descarta y se notifica, y la máquina funciona sin política hasta que llega un conjunto más reciente. Ese estado queda marcado en cada registro en lugar de silenciarse, y lo mismo ocurre cuando nunca se ha obtenido ninguna política: la acción se permite y el registro dice por qué. OpenLatch nunca bloquea una acción porque falte una política.

El material secreto lo guarda el almacén de credenciales del sistema operativo cuando existe y, si no, un archivo que solo puede leer su propietario. Una credencial pasada en una línea de comandos se rechaza, porque otros procesos de la máquina pueden leer una línea de comandos.

Telemetría de producto e informes de fallos

La telemetría de producto trata del software de OpenLatch, no de los agentes de tu organización. Contiene nombres de comandos, duraciones, el agente, códigos de error y volúmenes, y nunca texto de comandos, rutas de archivo, contenido de archivos ni prompts. El consentimiento se pide en la instalación, se puede cambiar después y está desactivado por defecto en integración continua. Los informes de fallos contienen un mensaje de error depurado y nada más, y se envían salvo que los desactives.

Ambos viajan a `app.openlatch.ai` con el tráfico que el cliente necesita para funcionar, así que una regla de red no puede pararlos sin aislar el cliente de la consola. Los ajustes están en el cliente: rechaza la telemetría en la instalación o ejecuta `openlatch system telemetry disable`, y desactiva los informes de fallos con `OPENLATCH_CRASH_REPORTING=0`. Desactivados, el cliente se comporta exactamente igual que con ellos activos. [Privacidad y telemetría](#) los explica con más detalle.

Informar de una vulnerabilidad

Envía los avisos de seguridad a security@openlatch.ai. OpenLatch confirma la recepción en un plazo de dos días hábiles y trabaja el caso con quien lo comunicó.

En resumen

- Salen de la máquina los metadatos de las acciones, los comandos y rutas de archivo enmascarados, los veredictos y las anomalías. El texto de los prompts se queda en la máquina salvo que tu organización lo active, y las peticiones y respuestas de modelos nunca se envían.
- Un único directorio contiene la configuración y los registros locales, en texto claro, protegidos por los permisos de archivo del usuario y tu cifrado de disco. Las credenciales las guarda el sistema operativo, o un archivo solo para su propietario.
- Los registros de acciones y decisiones se conservan 180 días, los agregados de uso 30 días, los archivos con hash y los registros de coste 90 días. El registro de auditoría es el único periodo que defines, con purga automática, y se exporta en CSV o JSON.
- La consola está alojada en Estados Unidos (US-East), con los datos de cada organización aislados por la base de datos. SAML 2.0 por organización, cuatro roles de owner a viewer, credenciales de cliente que heredan el rol de quien las emite, y un registro de auditoría de solo anexo y encadenado por hash.
- Las actualizaciones se comprueban en integridad antes de instalarse, y las políticas de la máquina se comprueban antes de usarse y en cada carga. Una máquina sin política utilizable permite la acción y lo indica en el registro.

Preguntas frecuentes del despliegue

Usa esta página para responder a las preguntas que plantea un despliegue. Cuando otra página trata una pregunta a fondo, la respuesta remite a ella.

Instalación

Pregunta	Respuesta
¿Qué necesitan de nosotros para empezar?	Tres cosas: una organización en la consola con un administrador que pueda invitar a todos los demás, HTTPS saliente en el 443 hacia <code>app.openlatch.ai</code> , y el perímetro, es decir, qué máquinas van en la primera oleada. Los scripts de instalación traen el cliente consigo, así que una máquina no necesita ningún entorno de ejecución. Requisitos previos lo convierte en una lista de comprobación
¿Necesita derechos de administrador en la máquina?	No. El cliente se instala y se ejecuta a nivel de usuario, no carga ningún controlador y no abre ningún puerto entrante
¿Podemos desplegarlo con nuestras herramientas de gestión actuales?	Sí. Como el cliente se ejecuta en contexto de usuario, se entrega como un script a nivel de usuario y no como un instalador a nivel de dispositivo, con la credencial pasada desde tu almacén de secretos por el entorno y nunca escrita en un script, un paquete ni una línea de comandos. Crea una clave por oleada de despliegue y revócala cuando termine la oleada. Consulta Desplegar el cliente
¿Qué agentes cubre?	La instalación detecta los agentes presentes en una máquina y conecta cada uno que encuentra. Niveles de control enumera los agentes conectados hoy, plano por plano. Cada agente aparece en la consola con su nivel de control, Enforced, Monitored o Known, que indica cuánto control tienes sobre él
¿Tiene que hacer algo el desarrollador?	Normalmente no. Algunos agentes piden al desarrollador que confirme la conexión una vez, la primera vez que se ejecutan tras la instalación, y <code>openlatch status</code> muestra qué agentes siguen pendientes. Consulta Verificar y desplegar

Red y proxy

Pregunta	Respuesta
¿A qué necesita llegar OpenLatch y abre algo?	Un único destino, saliente: <code>app.openlatch.ai</code> en TCP 443 por HTTPS, permitido por nombre porque no hay ningún rango de direcciones estático. Nada de entrada: el cliente solo escucha en la interfaz de bucle local. Consulta Requisitos de red
Nuestra única salida es un proxy con autenticación. ¿Qué se admite?	HTTP y HTTPS CONNECT, SOCKS5, proxies transparentes, y PAC o WPAD donde lo evalúa el resolvidor del sistema operativo. Se admiten las autenticaciones Basic y Kerberos (Negotiate). NTLM no, por diseño; Negotiate es la vía en esos entornos. Las credenciales de proxy se guardan en el almacén de credenciales del sistema operativo
¿La inspección TLS lo rompe?	No, siempre que la autoridad de certificación de interceptación sea de confianza. Instálala en el almacén de confianza del sistema operativo, o indícasela al cliente, en cuyo caso se combina encima del almacén del sistema. La verificación nunca se desactiva, y OpenLatch no emite ningún certificado propio
¿Cambia el tráfico propio de nuestros agentes?	No. Tus agentes llegan a sus puntos de acceso de modelos exactamente como antes. Esos puntos de acceso deben seguir siendo accesibles y quedar exentos del almacenamiento en búfer de respuestas y de la inspección de flujos
¿Qué pasa cuando una máquina está sin conexión, o la consola no es accesible?	Las políticas siguen vigentes en la máquina, porque la decisión se toma localmente. Los registros se guardan en la máquina y se envían cuando la consola vuelve a ser accesible. Consulta Arquitectura

Datos y seguridad

Pregunta	Respuesta
¿Es un proxy o una pasarela delante de nuestros agentes?	Ninguna de las dos cosas. OpenLatch se ejecuta dentro del propio ciclo de vida del agente, y la decisión se toma en la máquina antes de que la acción se ejecute. Nada nuestro está en tu camino de red, y OpenLatch no termina ninguna conexión TLS
¿Qué sale de la máquina? ¿Nuestros prompts? ¿Nuestro código fuente?	Los prompts se capturan como metadatos por defecto: que hubo uno, cuándo, en qué sesión y su longitud. El texto del prompt no se envía salvo que tu organización lo active, y las peticiones y respuestas intercambiadas con un modelo nunca se envían a OpenLatch. Lo que sale son los metadatos de las acciones con las credenciales y las rutas de archivo enmascaradas, más las señales de identidad necesarias para atribuir una acción; el contenido de los archivos se filtra por defecto. OpenLatch no se conecta a ninguno de tus repositorios. Consulta Tratamiento de datos y seguridad
¿Dónde está alojada la consola?	En Estados Unidos, en la región US-East, con los datos de cada organización aislados de los de todas las demás. Los veredictos se producen en tus propias máquinas, así que el lugar donde se guardan los registros no afecta a la aplicación. La lista de subencargados está disponible bajo petición
¿Quién puede iniciar sesión en la consola y quién ve qué?	El inicio de sesión único SAML 2.0 se configura por organización, y el acceso dentro de la consola sigue cuatro roles. Cada cambio administrativo se escribe en un registro de auditoría de solo anexo que puedes exportar. Consulta Configurar la organización
¿Puede alguien desactivarlo en su propia máquina?	OpenLatch se dirige a un agente que se comporta mal: que entra en bucle, malgasta presupuesto, sufre una inyección y actúa con sus herramientas, o incumple una política aunque por lo demás funcione con normalidad. El cliente se instala a nivel del propio usuario, así que una persona con control sobre su máquina puede quitarlo. Lo que pierde es la cobertura, no la discreción: la cobertura se informa máquina a máquina, y una máquina que deja de enviar datos aparece como un hueco en lugar de pasar por cubierta

Operación

Pregunta	Respuesta
¿Ralentiza al agente?	La decisión se toma en la máquina, sin llamada de red ni modelo en el camino, así que una acción cubierta no aguarda ningún viaje de ida y vuelta. OpenLatch no publica ninguna cifra de latencia. Lo que afirma es que cada acción cubierta llega al punto de decisión de las políticas, y que el veredicto producido por una política explícita es el que se aplica
¿Cómo confirmamos que una máquina está realmente conectada?	<code>openlatch status</code> indica si la máquina está conectada, si hay políticas presentes y si los agentes están conectados, y <code>openlatch doctor</code> demuestra cada uno de esos puntos. Después, los agentes de la máquina aparecen en el registro de agentes de la consola con un propietario. Consulta Verificar y desplegar
¿Cómo funcionan las actualizaciones y podemos fijarlas?	El cliente puede actualizarse, y cada descarga se comprueba en integridad antes de sustituir nada. Déjalo activado en las instalaciones de los desarrolladores. Un entorno gestionado lo desactiva, deja activada la comprobación y entrega la nueva versión en su propia ventana de mantenimiento. Consulta Operar
¿Cómo llevamos las notificaciones y los registros a nuestras herramientas?	Las notificaciones van por correo, a un canal de mensajería como Slack o Microsoft Teams, o a un webhook tuyo, y el registro de auditoría se exporta en CSV o JSON según la conservación que definas. Consulta Configurar los canales de notificación
¿Cómo lo desinstalamos?	<code>openlatch uninstall</code> desconecta el cliente de cada agente de la máquina y deja cada uno como estaba. Añadir <code>--purge</code> elimina además el directorio local y la credencial guardada. Consulta Operar
¿Cómo abrimos un caso de soporte?	Contacta con tu responsable técnico de OpenLatch por el canal acordado con tu organización. Ejecuta <code>openlatch doctor --rescue</code> en la máquina afectada y adjunta el archivo depurado que genera, con el sistema operativo y la hora del problema con su zona horaria. Los avisos de seguridad van a security@openlatch.ai

Términos y tráfico saliente

Usa esta página para consultar un término de la sección Desplegar, o para copiar la regla de salida en una solicitud de cambio.

Términos y siglas

Término	Significado
Agent Autonomy Zone	El perímetro operativo en el que trabaja un agente, expresado como políticas de intención en cinco dimensiones: Calidad, Economía, Seguridad, Protección y Cumplimiento. Consulta Autonomy Zone
Agent Intent Policy (AIP)	La forma en que se redacta en la consola la intención de tu organización, compilada en las reglas deterministas que deciden una acción
Veredictos	Allow, Ask, Block u Optimize, en este orden. Allow deja que la acción continúe. Ask la deja continuar y la marca para revisión asíncrona. Block la detiene antes de ejecutarse. Optimize cambia el camino: en Enforce, una palanca actúa, y en Monitor, mide lo que habría hecho. Consulta Veredictos
Cliente	El software de OpenLatch instalado una vez en cada máquina que ejecuta un agente. Decide una acción cubierta, en la máquina, antes de que se ejecute
Consola	La aplicación web de OpenLatch en <code>app.openlatch.ai</code> , donde configuras la organización, redactas las políticas y ves lo que hicieron tus agentes
Nivel de control	Cuánto control tiene OpenLatch sobre un agente. Enforced: el veredicto llega al agente y se cumple. Monitored: los eventos fluyen, pero no se puede entregar ningún veredicto. Known: el agente está identificado y clasificado, con los límites del control indicados. Consulta Niveles de control
IdP	Proveedor de identidad: el directorio y el servicio de autenticación de tu organización
MDM	Gestión de dispositivos móviles: las herramientas que entregan software y ajustes a las máquinas gestionadas
PAC / WPAD	Configuración automática de proxy y Web Proxy Auto-Discovery: un script que indica a un cliente qué proxy usar, y el mecanismo que lo localiza
SAML2.0	Security Assertion Markup Language, versión 2.0: el protocolo de inicio de sesión único que la consola admite por organización
SNI	Server Name Indication: la extensión TLS que lleva en claro el nombre del host de destino, que es lo que lee una autorización por nombre
Inspección TLS	Un proxy que termina y vuelve a emitir TLS para inspeccionar el tráfico. Se admite cuando la autoridad de certificación de interceptación es de confianza

Término	Significado
Oleada	Una etapa de un despliegue. La sección Desplegar tiene tres oleadas de tamaño creciente

Tráfico saliente requerido

Entrega esta tabla a quien gestione tu tráfico saliente. Una máquina que ejecuta el cliente necesita exactamente un destino de OpenLatch, y la telemetría de producto y los informes de fallos opcionales también viajan a él. La segunda fila solo es necesaria mientras instalas o actualizas el cliente. La tercera es para los navegadores que inician sesión en la consola; ninguna máquina que ejecuta el cliente la contacta.

Destino	Puerto	Sentido	Protocolo	Cómo permitirlo
<code>app.openlatch.ai</code>	TCP 443	Saliente	HTTPS	Por nombre. No hay ningún rango de direcciones estático. Esta entrada lleva también la telemetría de producto y los informes de fallos opcionales
Tu registro de paquetes: <code>registry.npmjs.org</code> , salvo que apuntes el cliente a un espejo	TCP 443	Saliente	HTTPS	Por nombre, o usa tu espejo interno. Necesario para una instalación o una actualización, no para funcionar
<code>challenges.cloudflare.com</code> , desde el navegador de cualquier persona que inicie sesión en la consola	TCP 443	Saliente	HTTPS	Por nombre, solo para navegadores. Sirve la comprobación que hace el paso de correo al iniciar sesión

Tus agentes llegan a sus puntos de acceso de modelos exactamente como antes. Esos puntos de acceso deben seguir siendo accesibles y exentos del almacenamiento en búfer de respuestas y de la inspección de flujos.

No se necesita ningún puerto entrante, ni en la máquina ni en ningún otro sitio. OpenLatch no emite certificados ni los exige. [Requisitos de red](#) trata el tema a fondo, incluidos los proxies, la autenticación y la inspección TLS.

Referencia de AIP

Usa esta página para leer o revisar una Agent Intent Policy (AIP): de qué está hecha, qué dimensión de la Autonomy Zone cubre, en qué modo se ejecuta y qué veredicto devuelve para una acción cubierta.

Una política de intención expresa lo que quiere decir tu organización, en lugar de cada superficie que un agente podría usar. OpenLatch escribe cada política de intención como una AIP y la compila en reglas deterministas que el cliente aplica antes de que se ejecute la acción.

Átomos de intención

Una AIP es un conjunto de uno o más **átomos de intención**. Una cláusula de tu documento de origen se convierte en un átomo, y todos los átomos tienen los mismos campos fijos.

Campo	Contenido
who	Los agentes a los que se aplica el átomo. Cuando el origen no lo dice, el átomo indica que es una deducción
does what	La acción, como un despliegue o la eliminación de un archivo
to what	El objetivo de la acción, como un espacio de nombres de producción
only if	Las condiciones en las que la acción es aceptable
unless	Las excepciones al átomo
class	Si el átomo se puede aplicar: <code>enforceable</code> , <code>monitor-only</code> o <code>out-of-scope</code>

A partir de estos campos, OpenLatch genera una **frase** y un conjunto de **casos de prueba**. Nunca editas las reglas compiladas.

Lo que ves	Para qué sirve
La frase	Lo que ratificas, con un solo Sí o No
El origen	La cláusula de tu documento de la que procede el átomo, citada para que compruebes que se leyó bien
Los casos de prueba	Situaciones concretas con el veredicto que obtendría cada una, mostradas en modo de solo lectura junto a la frase para que tu respuesta sea informada

Un No devuelve el átomo para corregirlo: dices lo que querías decir, los campos cambian y el átomo vuelve con una frase nueva y casos nuevos para una nueva respuesta. Cambiar un átomo siempre anula la ratificación que se dio a su versión anterior.

La `class` de un átomo es una propiedad del átomo, no un ajuste. Un átomo `monitor-only` describe algo que se puede observar pero no detener antes de que ocurra, así que permanece en Monitor aunque su política esté en Enforce.

Dimensiones

Cada átomo pertenece a una de las cinco dimensiones de la Autonomy Zone. Una política cubre la unión de las dimensiones de sus átomos.

Dimensión	La pregunta que responde	Ejemplos de políticas de intención
Calidad	¿Está el trabajo a la altura?	El código se prueba antes de publicarse; se sigue el proceso obligatorio
Economía	¿Está justificado el coste?	Sin bucles descontrolados ni reintentos inútiles; el modelo se ajusta a la tarea
Seguridad	¿Puede tocar esto?	Ninguna credencial sale del perímetro de la empresa; solo herramientas y servidores aprobados
Protección	¿Es aceptable la consecuencia?	Sin acceso a producción sin aprobación; sin operaciones irreversibles o destructivas
Cumplimiento	¿Es legal y conforme a las normas?	Políticas y normas internas; obligaciones contractuales

La misión no es una dimensión: cada agente aporta su propia misión, y la zona aporta las reglas que debe seguir.

Modos

Una política se ejecuta en uno de tres modos. El modo pertenece a la política, nunca a un átomo, y se define en cada Autonomy Zone que contiene la política.

Modo	Qué ocurre cuando coincide
Disabled	Nada. La política y su historial se conservan, pero no se evalúa
Monitor	La coincidencia se registra como anomalía y se marca; la acción continúa
Enforce	El veredicto se aplica antes de que se ejecute la acción, allí donde un adaptador cubre esa acción, y el registro indica si realmente se aplicó

Cuando las zonas de un agente contienen la misma política más de una vez, se ejecuta una sola vez, en el modo más estricto: Enforce por delante de Monitor, y Monitor por delante de Disabled.

Las políticas se ganan el modo Enforce con evidencias: ejecuta una política en Monitor, revisa sus anomalías y pásala a Enforce cuando sea precisa. OpenLatch nunca amplía por sí mismo la autoridad de un agente.

Veredictos

Cada acción cubierta que coincide con una política en Enforce recibe uno de cuatro veredictos antes de ejecutarse. Un veredicto es determinista porque lo ha producido una política explícita.

Veredicto	Qué le ocurre a la acción
Allow	La acción continúa sin cambios. En Monitor, puede permitirse y marcarse
Ask	La acción se permite y se marca para revisión de forma predeterminada
Block	La acción se impide antes de ejecutarse. El agente recibe el motivo y puede intentar otra cosa; la sesión continúa
Optimize	La acción toma otro camino, mediante una de las palancas siguientes

Ask

Ask permite la acción y la marca para una revisión asíncrona de forma predeterminada. La decisión de la revisión, con el alcance que elige el revisor, llega al cliente en el siguiente paquete de políticas, así que la siguiente acción similar se decide en la máquina, sin ninguna persona.

Ask solo retiene una acción y espera una aprobación con un opt-in por átomo para una acción irreversible, como eliminar una base de datos de producción.

Codex CLI no tiene una solicitud de confirmación nativa. En Codex CLI, Ask le llega al agente como un mensaje con el motivo de la regla, y la llamada continúa.

Ask evita que se repita, no la primera vez. Escribe como Block todo lo que nunca deba ocurrir sin revisión, o marca la acción como irreversible en su átomo.

Palancas de Optimize

Optimize cambia el camino de una acción en lugar de permitirla o bloquearla. El modo decide lo que hace una palanca: en Enforce una palanca actúa, y en Monitor una palanca solo mide lo que habría hecho.

Palanca	Qué cambia	Estado
loop_stop	Detiene a un agente que repite la misma llamada y le dice por qué	Actúa en Enforce
steer	Rechaza la acción y le dice al agente qué hacer en su lugar, como subir una rama en lugar de la rama principal	Actúa en Enforce
narrow_output	Devuelve al agente un extracto filtrado de una salida de comando larga, y guarda la salida completa en el disco	Actúa en Enforce, en Claude Code
effort_clamp	Reduce el esfuerzo de razonamiento de una solicitud al modelo	Mide en Monitor y en Enforce; actúa en Enforce solo donde la organización cliente lo activa
context_edit	Ajusta la configuración de limpieza del contexto de una solicitud al modelo	Mide en Monitor y en Enforce; actúa en Enforce solo donde la organización cliente lo activa
prefix_guard	Mantiene estable el prefijo de prompt en caché de una solicitud al modelo	Mide en Monitor y en Enforce; actúa en Enforce solo donde la organización cliente lo activa
substitute	Sustituye un comando por un equivalente que aprobó tu organización	No está publicado: no tiene actuador y solo mide en Monitor

En Enforce, loop_stop, steer y narrow_output actúan (narrow_output en Claude Code). effort_clamp, context_edit y prefix_guard solo miden, incluso en Enforce, salvo que la organización cliente lo active. substitute no está publicado, así que solo mide en Monitor. En Monitor, todas las palancas solo miden.

El registro de evidencias indica cuál se aplica: aplicado en Enforce, medido en Monitor.

Qué cubre un veredicto

Cada acción cubierta llega al punto de decisión de las políticas, y su veredicto se aplica. Una acción fuera de una superficie cubierta, como un script que el agente escribió antes y ejecuta después, no la decide el cliente. El nivel de control de cada agente indica qué cobertura se le aplica.

Configuración del cliente

Usa esta página para cambiar cómo se comporta el cliente en una máquina: dónde guarda su estado, cómo llega a la plataforma y cuáles de sus partes se ejecutan.

`openlatch init` escribe la configuración por ti. Edítala a mano solo para un ajuste que ningún comando cubre, y después ejecuta `openlatch restart` para aplicarla: el daemon solo lee su configuración una vez, al arrancar.

De dónde vienen los ajustes

El cliente lee cada ajuste de la primera fuente que lo define, en este orden:

1. Una opción en la línea de comandos.
2. Una variable de entorno (`OPENLATCH_*`).
3. El archivo de configuración, `config.toml` en el directorio de estado: `~/ .openlatch` en macOS y Linux, `%APPDATA%\openlatch` en Windows.
4. El valor predeterminado integrado.

La ruta del proxy es la única excepción: se resuelve clave por clave y, después del archivo de configuración, también tiene en cuenta las variables estándar de la familia `https_proxy`, y luego los ajustes de proxy del sistema operativo, antes de ir en directo.

El daemon supervisado no hereda las variables de tu shell. Todo lo que necesita, como la dirección de la plataforma, `openlatch init` lo escribe en `config.toml` para que el daemon también lo use.

Variables de entorno

Estado y conexión

Variable	Qué define	Predeterminado
OPENLATCH_DIR	El directorio de estado: configuración, credencial, registros y outbox	~/ .openlatch en macOS y Linux, %APPDATA%\openlatch en Windows
OPENLATCH_PORT	El puerto del daemon. Si está definida, <code>openlatch init</code> usa exactamente este puerto; si no, busca uno libre del 7443 al 7543	7443
OPENLATCH_API_URL	La dirección de la plataforma a la que el cliente envía los eventos y de la que obtiene las políticas	<code>https://app.openlatch.ai</code>
OPENLATCH_API_KEY	La credencial de registro. <code>openlatch init</code> la guarda en el llavero del sistema operativo, o en un archivo cifrado cuando no hay ningún llavero utilizable	—
OPENLATCH_SKIP_KEYRING	Considera el llavero no disponible, para que prevalezca la variable de entorno	sin definir
OPENLATCH_LOG	Nivel de registro: <code>error</code> , <code>warn</code> , <code>info</code> , <code>debug</code> o <code>trace</code>	<code>info</code>

Proxy

Variable	Qué define	Predeterminado
OPENLATCH_PROXY	La URL del proxy para cada solicitud saliente. Es el único ajuste que puede llevar una credencial; <code>openlatch init</code> mueve la contraseña al llavero en lugar de escribirla en el disco	sin definir
OPENLATCH_NO_PROXY	Destinos adicionales que no pasan por el proxy. Las direcciones de bucle local nunca pasan por él	sin definir
OPENLATCH_PROXY_MODE	<code>auto</code> , <code>manual</code> o <code>direct</code>	<code>auto</code>
OPENLATCH_PROXY_AUTH	<code>auto</code> , <code>none</code> , <code>basic</code> o <code>negotiate</code> (Kerberos). NTLM no es compatible	<code>auto</code>
OPENLATCH_PROXY_PAC_URL	Una URL explícita de configuración automática del proxy, usada en macOS y Windows. Se rechaza en Linux con 0L-1225	sin definir
OPENLATCH_PROXY_SPN	El nombre de servicio Kerberos del proxy	HTTP/<proxy-host>
OPENLATCH_CA_BUNDLE	Un paquete PEM que se añade a los certificados de confianza del sistema operativo, para un proxy que inspecciona el tráfico TLS	sin definir

Partes del cliente

Variable	Qué define	Predeterminado
OPENLATCH_POLICY_ENABLED	La evaluación local de políticas. <code>false</code> desactiva por completo la aplicación en esta máquina; no es un modo Monitor (OL-1215)	<code>true</code>
OPENLATCH_MODEL_RELAY_ENABLED	El relé de modelo, el receptor local que registra las llamadas al modelo de cada agente. El daemon solo dirige un agente hacia él cuando una solicitud de prueba demuestra que llega al proveedor	<code>true</code>
OPENLATCH_MODEL_RELAY_PORT	El puerto del relé de modelo	<code>7600</code>
OPENLATCH_INVENTORY_ENABLED	El seguimiento de los archivos de configuración de los agentes (el plano de configuración)	<code>true</code>
OPENLATCH_CLINE_PLUGIN_DELIVERY	La entrega del plugin de OpenLatch a Cline.app y a la extensión de Cline para VS Code. Equivale a <code>plugin_delivery</code> en <code>[cline]</code>	<code>true</code>

Envío a la plataforma

Variable	Qué define	Predeterminado
OPENLATCH_CLOUD_BATCH_MAX_EVENTS	Eventos enviados por lote, de 1 a 100	50
OPENLATCH_CLOUD_BATCH_MAX_WAIT_MS	Cuánto espera el primer evento de un lote antes de que se envíe el lote, en milisegundos	5000
OPENLATCH_CLOUD_OUTBOX_MAX_BYTES	El volumen máximo de eventos no enviados que guarda el outbox, en bytes, antes de descartar los más antiguos. 0 quita el límite	104857600
OPENLATCH_CLOUD_FALLBACK_MAX_BYTES	El volumen máximo de eventos no reproducidos que guarda el registro de respaldo, en bytes, antes de omitir los más antiguos. Limita esa ventana, no el archivo: durante un corte largo, el archivo en disco sigue creciendo hasta que el daemon lo reproduce. Solo lo lee el daemon; el hook mantiene su propio límite de 50 MB. 0 quita el límite	52428800

El envío de eventos a la plataforma no se puede desactivar. Para enviarlos a otro lugar, cambia OPENLATCH_API_URL.

Actualizaciones y telemetría

Variable	Qué define	Predeterminado
OPENLATCH_AUTO_UPDATE	Las actualizaciones automáticas: el daemon busca periódicamente una nueva versión y la instala cuando no se está ejecutando ningún hook. <code>true</code> o <code>1</code> las activa; cualquier otro valor las desactiva. Equivale a <code>auto_update</code> en <code>[update]</code>	<code>true</code>
OPENLATCH_UPDATE_CHECK	La comprobación de versión al arrancar. <code>false</code> o <code>0</code> la desactiva. Equivale a <code>check</code> en <code>[update]</code>	<code>true</code>
OPENLATCH_NPM_REGISTRY	El registro desde el que descarga <code>openlatch update</code>	<code>https://registry.npmjs.org</code>
OPENLATCH_TELEMETRY_DISABLED	<code>1</code> desactiva la telemetría de uso	sin definir
DO_NOT_TRACK	<code>1</code> desactiva la telemetría de uso, como en otras herramientas que siguen esta convención	sin definir
OPENLATCH_CRASH_REPORTING	<code>0</code> , <code>false</code> , <code>no</code> u <code>off</code> detiene los informes de fallos, con independencia de la telemetría de uso	activado

Las actualizaciones automáticas nunca se ejecutan en runners de CI.

Archivo de configuración

`config.toml` en el directorio de estado contiene los ajustes que escriben los comandos, agrupados en secciones. Las secciones que es más probable que leas o edites:

Sección	Clave	Qué define
[policy]	enabled	La evaluación local de políticas, como OPENLATCH_POLICY_ENABLED
[model_relay]	enabled	El relé de modelo, como OPENLATCH_MODEL_RELAY_ENABLED. openlatch system model-relay enable y disable lo escriben
[cline]	plugin_delivery	La entrega del plugin de OpenLatch a los hosts de Cline
[update]	auto_update · check	Las actualizaciones automáticas y la comprobación de versión al arrancar
[proxy]	ver más abajo	La ruta que sigue cada solicitud saliente

La sección [proxy]

openlatch init encuentra la ruta por sí solo cuando puede y la escribe aquí. openlatch system proxy set, discover y clear la cambian; úsalos en lugar de editar el archivo.

```
[proxy]
mode = "auto"
url = "http://proxy.corp:8080"
auth = "auto"
no_proxy = ".corp.example,10.0.0.0/8"
ca_bundle = "/etc/ssl/corp-root.pem"
allow_direct = true
```

Clave	Qué define
<code>mode</code>	<code>auto</code> busca una ruta que funcione, <code>manual</code> usa exactamente lo definido y nunca se sustituye, <code>direct</code> nunca usa un proxy
<code>url</code>	El proxy, como <code>http://</code> , <code>https://</code> , <code>socks5://</code> o <code>socks5h://</code> . Nunca contiene un nombre de usuario ni una contraseña
<code>username</code>	La cuenta con la que autenticarse. La contraseña está en el llavero, nunca en este archivo
<code>auth</code>	<code>auto</code> elige un esquema cuando arranca el daemon; <code>basic</code> , <code>negotiate</code> y <code>none</code> son explícitos
<code>no_proxy</code>	Destinos adicionales que no pasan por el proxy, como <code>example.com</code> , <code>.example.com</code> o <code>10.0.0.0/8</code>
<code>ca_bundle</code>	Un paquete PEM que se añade a los certificados de confianza, nunca en su lugar
<code>allow_direct</code>	<code>false</code> rechaza una conexión directa cuando no se puede usar ningún proxy (0L-1227)
<code>spn</code>	El nombre de servicio Kerberos del proxy
<code>http1_only</code>	Fuerza HTTP/1.1, para un proxy de inspección que gestiona mal HTTP/2
<code>pac_url</code>	Una URL explícita de configuración automática del proxy. Se rechaza en Linux (0L-1225)

Ejecuta `openlatch system proxy test` después de cada cambio para comprobar la ruta.

Lo que el cliente escribe en los ajustes de los agentes

Mientras el relé de modelo está activado, el daemon dirige hacia él las llamadas al modelo de cada agente detectado, y quita esa conexión cuando detienes el daemon, desactivas el relé o desinstalas. Un valor que hayas definido tú nunca se sobrescribe: `openlatch doctor` lo indica en su lugar (0L-RELAY-SETTINGS).

Agente	Dónde va la conexión
Claude Code	Claves de entorno en sus ajustes: la dirección del relé, un identificador de instalación y entradas de bucle local en NO_PROXY
Codex CLI	Una entrada de proveedor openlatch en ~/.codex/config.toml
Cursor	La dirección del relé en los ajustes de usuario del IDE, y un wrapper de shell para la CLI de Cursor que recoge un terminal nuevo
Cline	Un punto de conexión local del relé por cada proveedor configurado, en los archivos de ajustes de Cline

Conectar Claude Code al relé de modelo define su ANTHROPIC_BASE_URL, lo que desactiva Claude Code Remote Control mientras el relé está activado. En una máquina detrás de un proxy, exporta también NO_PROXY=127.0.0.1, localhost en el shell que inicia Codex CLI, para que sus llamadas al modelo lleguen al relé local en lugar de al proxy corporativo.

Ejecuta openlatch system model-relay status para ver qué agentes tienen sus llamadas al modelo pasando por el relé.

Glosario

Usa esta página cuando un término de la consola, de la salida del cliente o de esta documentación necesite un significado preciso. Los nombres de producto, los veredictos y los modos conservan su forma en inglés en todos los idiomas.

Autonomy Zone y políticas

Término	Significado
Agent Autonomy Zone	El marco de actuación que una organización define para sus agentes de IA: qué puede hacer un agente, en qué condiciones y con qué consecuencias aceptadas. Forma corta: Autonomy Zone
Dimensión	Uno de los cinco lados de la Autonomy Zone: Calidad, Economía, Seguridad, Protección y Cumplimiento
Zona efectiva	La única zona que corresponde a un agente. Una unidad de tu organización sin zona propia hereda la más cercana por encima de ella
Política de intención	Una regla que expresa lo que quiere decir tu organización, en lugar de cada superficie que un agente podría usar
Agent Intent Policy (AIP)	La forma que toma una política de intención en OpenLatch: un conjunto de átomos de intención, compilado en reglas deterministas. Consulta la referencia de AIP
Átomo de intención	Una cláusula de una política, guardada en campos fijos (who, does what, to what, only if, unless) a partir de los cuales se generan una frase y casos de prueba
Ratificación	El Sí o el No del responsable de la política sobre la frase de un átomo, con sus casos de prueba mostrados al lado
Modo	Cómo se ejecuta una política en una zona: Disabled, Monitor o Enforce
Disabled	Un modo en el que la política se conserva pero no se evalúa
Monitor	Un modo en el que una coincidencia se registra y se marca, y la acción continúa
Enforce	Un modo en el que el veredicto se aplica antes de que se ejecute la acción, allí donde un adaptador la cubre

Decisiones

Término	Significado
Veredicto	La decisión que una política devuelve para una acción cubierta, antes de ejecutarse: Allow, Ask, Block u Optimize
Allow	La acción continúa sin cambios
Ask	La acción se permite y se marca para revisión de forma predeterminada. Solo se retiene para aprobación con un opt-in por átomo para una acción irreversible
Block	La acción se impide antes de ejecutarse, y el agente recibe el motivo
Optimize	La acción toma otro camino, mediante una palanca de Optimize
Palanca de Optimize	Una forma en que Optimize cambia una acción. En Enforce, una palanca actúa donde está publicada; en Monitor, solo mide lo que habría hecho
Anomalía	Una acción de un agente que infringe una política de intención de una de las dimensiones de la zona
Determinista	Se dice de un veredicto producido por una política explícita: la misma acción en el mismo contexto obtiene siempre el mismo veredicto

Agentes y cobertura

Término	Significado
Agente autorizado	Un agente que tu organización ha aprobado y conectado a OpenLatch
Nivel de control	Cuánto controla OpenLatch a un agente: Enforced, Monitored o Known
Enforced	Un nivel de control: un adaptador de OpenLatch está en la ruta de ejecución del agente y aplica los veredictos
Monitored	Un nivel de control: las acciones del agente son visibles y sus anomalías se registran, pero los veredictos no se aplican
Known	Un nivel de control: el agente está identificado y clasificado, y se indican los límites del control
Plano de comandos	Las acciones que un agente realiza con sus herramientas: comandos, cambios en archivos, llamadas a herramientas y servidores
Plano de solicitudes	Las solicitudes que un agente envía a su proveedor de modelo
Grafo de ejecución	El registro de lo que hizo un agente en una sesión: llamadas al modelo, llamadas a herramientas, comandos, cambios en archivos, acceso a datos y coste, en orden
Registro de evidencias	Lo que OpenLatch conserva de cada decisión: la acción, la política, el modo, el veredicto y si se aplicó

El cliente

Término	Significado
Cliente	El programa de OpenLatch instalado en cada máquina donde se ejecutan agentes
Daemon	El proceso en segundo plano del cliente, que recibe cada evento de un agente y lo decide
Hook	El punto de entrada al que llama un agente antes y después de cada acción, y que pasa la acción al daemon
Paquete de políticas	El archivo de reglas con integridad comprobada que el cliente obtiene de la plataforma y guarda en el disco, para seguir decidiendo con la red desconectada
Relé de modelo	Un receptor local hacia el que el cliente dirige las llamadas al modelo de cada agente, para que se registre el plano de solicitudes
Outbox	La cola local donde el cliente guarda los eventos hasta que la plataforma los ha recibido
Supervisor	El servicio del sistema operativo (launchd, systemd o el Programador de tareas) que inicia el daemon al iniciar sesión y lo reinicia
Plano de configuración	Los archivos de configuración de los agentes cuyos cambios sigue el cliente
Código de error	Un código como 0L-1002 que el cliente muestra junto a un problema. Cada uno tiene su propia página, enumerada en la referencia de la CLI

La plataforma

Término	Significado
Plataforma	El servicio de OpenLatch donde defines zonas y políticas y revisas lo que hicieron los agentes, accesible en la consola en <code>app.openlatch.ai</code>
Consola	La interfaz web de la plataforma
Organización	La cuenta de tu empresa en la plataforma, que contiene sus unidades, zonas, políticas y agentes
Clave de API	Una credencial que permite al cliente o a un script actuar en nombre de tu organización en la plataforma

Referencia de la CLI

Todos los comandos y subcomandos del cliente OpenLatch.

openlatch init

`openlatch setup`

Instala los hooks de los agentes encontrados en esta máquina, inicia el daemon y comprueba que responde. Se puede ejecutar de nuevo sin riesgo: primero libera los puertos del cliente y después confirma que el daemon que inició es el que responde.

openlatch status

Muestra el estado del daemon, su tiempo de actividad y el número de eventos, el estado de cada parte del cliente y la línea de licencia: su estado, los hosts, el uso razonable y el fin del contrato.

openlatch start

Inicia el daemon. Se niega con OL-1501 cuando ya hay un daemon en marcha en esta máquina.

openlatch stop

Detiene el daemon. También quita la conexión de los agentes con el relé de modelo, y lo indica.

openlatch restart

Detiene el daemon y lo vuelve a iniciar, por ejemplo para aplicar un cambio de configuración.

openlatch logs

Muestra el registro de eventos del cliente.

openlatch doctor

Diagnostica la configuración y la conectividad del cliente, e indica la causa y la solución de cada problema que encuentra. `--fix` aplica las reparaciones posibles, `--restore` las deshace y `--rescue` genera un archivo de diagnóstico para el soporte.

openlatch uninstall

Quita los hooks de los agentes y detiene el daemon. `--agent` lo limita a los agentes que indiques.

openlatch docs

Abre la documentación en tu navegador.

openlatch system

Agrupar los comandos que inspeccionan o controlan una parte del cliente: `aip`, `auth`, `hooks`, `model-relay`, `proxy`, `supervision`, `telemetry` e `inventory`. Cada solución que da `openlatch doctor` indica el comando `system` exacto que hay que ejecutar.

openlatch system aip

Obtiene e inspecciona el paquete de políticas presente en esta máquina, que contiene las reglas de la Autonomy Zone que aplica el cliente.

openlatch system aip refresh

```
openlatch system aip update
```

Hace que el daemon en marcha obtenga el paquete de políticas ahora mismo, y espera el resultado hasta 15 segundos: cambiado, sin cambios, pendiente, rechazado o sin conexión, con los datos del paquete en uso. Un resultado pendiente termina con el código 7; un rechazo o un daemon sin conexión termina con el código 1.

openlatch system auth

Agrupar los comandos que conectan el cliente con la plataforma.

openlatch system auth login

Inicia sesión del cliente en la plataforma a través de tu navegador y guarda la credencial en el llavero del sistema operativo.

openlatch system auth logout

Cierra la sesión del cliente y elimina la credencial guardada.

openlatch system auth status

Indica si el cliente ha iniciado sesión y de dónde procede su credencial.

openlatch system auth switch

Mueve esta máquina a otra de tus organizaciones.

openlatch system hooks

Agrupar los comandos que gestionan los hooks de los agentes. `system hooks install` equivale a `openlatch init`.

openlatch system hooks install

Instala los hooks de los agentes; equivale a `openlatch init`.

openlatch system hooks uninstall

Quita los hooks de los agentes; equivale a `openlatch uninstall`.

openlatch system hooks status

Muestra en detalle los hooks instalados para cada agente.

openlatch system model-relay

Agrupar los comandos que inspeccionan o activan y desactivan el relé de modelo, el receptor local que registra las llamadas al modelo de cada agente.

openlatch system model-relay status

Indica si el relé de modelo está en marcha, su puerto y si las llamadas al modelo de cada agente pasan por él. Un agente cuyos hooks funcionan pero cuyas llamadas al modelo no se registran aparece como aplicado, pero no medido.

openlatch system model-relay enable

Activa el relé de modelo en la configuración y ofrece reiniciar el daemon para aplicarlo.

openlatch system model-relay disable

Desactiva el relé de modelo en la configuración y ofrece reiniciar el daemon para aplicarlo.

openlatch system model-relay explain

Muestra un registro del relé de modelo por su identificador, leído del almacenamiento local. Su contenido nunca sale de la máquina.

openlatch system proxy

Agrupar los comandos que inspeccionan, descubren, definen o prueban el proxy por el que pasa cada solicitud saliente.

openlatch system proxy status

Muestra la ruta que siguen las solicitudes salientes, para la CLI y, si es distinta, para el daemon.

openlatch system proxy discover

Vuelve a ejecutar la detección de proxy del sistema operativo, prueba cada candidato y conserva el que funciona.

openlatch system proxy set

Define el proxy a mano. Un proxy definido así nunca lo sustituyen la detección ni la reparación automática.

openlatch system proxy clear

Quita el proxy y su credencial, para que las solicitudes vayan directamente a su destino.

openlatch system proxy test

Prueba la ruta paso a paso: la conexión al proxy, TLS, HTTP y una respuesta en streaming.

openlatch system supervision

Agrupar los comandos que gestionan el supervisor del sistema operativo (launchd, systemd o el Programador de tareas) que inicia el daemon al iniciar sesión y lo reinicia. Todo queda en el ámbito del usuario.

openlatch system supervision install

Instala el supervisor del sistema operativo, para que el daemon se inicie al iniciar sesión y tras un reinicio.

openlatch system supervision uninstall

Quita el supervisor del sistema operativo. El daemon sigue en marcha, pero ya no se inicia solo.

openlatch system supervision status

Indica si el supervisor está instalado y en marcha, y cuál usa este sistema.

openlatch system supervision enable

Vuelve a activar la supervisión después de `disable`, e instala de nuevo el supervisor.

openlatch system supervision disable

Quita el supervisor y registra que la supervisión está desactivada en esta máquina.

openlatch system telemetry

Agrupar los comandos que gestionan la telemetría de uso, que puedes desactivar.

openlatch system telemetry status

Indica si la telemetría de uso está activada, qué ajuste lo decide y si esta versión la incluye.

openlatch system telemetry enable

Activa la telemetría de uso.

openlatch system telemetry disable

Desactiva la telemetría de uso.

openlatch system telemetry purge

Desactiva la telemetría de uso y vacía la cola de eventos no enviados de un daemon en marcha.

openlatch system telemetry debug

Muestra cómo ejecutar cualquier comando para que imprima los eventos de telemetría que enviaría.

openlatch system inventory

Agrupar los comandos que inspeccionan el plano de configuración: las fuentes de configuración de los agentes que sigue el cliente, sus eventos de cambio, los nuevos análisis y el estado del observador.

openlatch system inventory list

Enumera las fuentes de configuración que sigue el cliente, a partir del registro de auditoría local.

openlatch system inventory log

Muestra los eventos de cambio de configuración del registro de auditoría local.

openlatch system inventory rescan

Vuelve a analizar ahora cada fuente de configuración seguida.

openlatch system inventory status

Indica el estado del observador de configuración: si está activado, si su lista de fuentes se cargó y cuántas fuentes contiene.

openlatch system inventory inspect

Muestra una fuente de configuración en detalle: su huella actual y sus alertas pendientes.

openlatch system inventory projects

Enumera las carpetas de proyecto que el cliente conoce por los archivos de configuración que sigue.

openlatch system inventory ack

Confirma una alerta de configuración pendiente, para que el agente deje de recibir el recordatorio.

openlatch update

Busca una nueva versión del cliente. En una terminal, pregunta después si instalarla, y la instala si aceptas. Sin terminal (un script, una tubería, una herramienta de gestión), solo informa de la versión. --yes instala sin preguntar; --check solo comprueba.

Códigos de error

Cada código que el cliente puede mostrar, con lo que ha pasado y qué hacer.

- [OL-1002 – Evento demasiado grande](#)
- [OL-1003 – Evento duplicado ignorado](#)
- [OL-1100 – La expresión personalizada no es válida](#)
- [OL-1200 – Plataforma inaccesible](#)
- [OL-1201 – La plataforma rechazó la credencial](#)
- [OL-1202 – Límite de frecuencia de la plataforma alcanzado](#)
- [OL-1204 – No se pudo escribir en el outbox](#)
- [OL-1205 – Outbox enviado solo en parte](#)
- [OL-1206 – Canal hacia la plataforma saturado](#)
- [OL-1207 – Entrada del outbox apartada](#)
- [OL-1208 – Evento retenido caducado](#)
- [OL-1210 – No se pudo obtener el paquete de políticas](#)
- [OL-1211 – El paquete de políticas no superó la verificación](#)
- [OL-1212 – El paquete de políticas no se puede usar](#)
- [OL-1213 – Actualización del paquete de políticas atrasada](#)
- [OL-1214 – Una regla del paquete quedó apartada](#)
- [OL-1215 – La política local está desactivada](#)
- [OL-1216 – La nueva versión no puede leer la caché de políticas](#)
- [OL-1220 – Sin ruta hacia la plataforma](#)
- [OL-1221 – Proxy inaccesible](#)

- [OL-1222 – El proxy rechazó la credencial](#)
- [OL-1223 – Esquema de proxy no admitido](#)
- [OL-1224 – Certificado TLS no fiable](#)
- [OL-1225 – Configuración automática del proxy no disponible](#)
- [OL-1226 – Configuración del proxy no válida](#)
- [OL-1227 – Conexión directa no permitida](#)
- [OL-1228 – El proxy almacena en búfer las respuestas en streaming](#)
- [OL-1229 – Sin ticket Kerberos para el proxy](#)
- [OL-1240 – Fila de entrada de evaluación mal formada](#)
- [OL-1241 – Paquete de evaluación rechazado](#)
- [OL-1242 – Ningún paquete con el que evaluar](#)
- [OL-1243 – El estado de sesión no coincide con el paquete](#)
- [OL-1244 – Falló la evaluación de una fila](#)
- [OL-1250 – Recepción pausada por motivos de licencia](#)
- [OL-1300 – Valor de configuración no válido](#)
- [OL-1301 – Falta un campo de configuración obligatorio](#)
- [OL-1302 – Configuración cambiada pero no aplicada](#)
- [OL-1400 – No se encontró ningún agente compatible](#)
- [OL-1401 – No se pueden escribir los ajustes del agente](#)
- [OL-1402 – No se puede analizar el archivo de ajustes del agente](#)
- [OL-1403 – Otros hooks en los ajustes del agente](#)
- [OL-1404 – No se puede instalar el binario del hook](#)
- [OL-1405 – Hooks instalados pero sin ejecutarse](#)
- [OL-1406 – No se puede analizar la configuración TOML del agente](#)
- [OL-1407 – El agente seleccionado no admite hooks](#)
- [OL-1408 – Falta un componente esperado de Cline](#)
- [OL-1409 – Estado de un componente de Cline desconocido](#)
- [OL-1410 – Las acciones de Cline no se pueden rechazar](#)
- [OL-1411 – Llamadas al modelo de Cline.app no capturadas](#)
- [OL-1412 – Plugin de OpenLatch no cargado en Cline](#)
- [OL-1413 – Formato desconocido del archivo de hooks de Cursor](#)
- [OL-1414 – No se puede confirmar la instalación de Cursor](#)

- [OL-1415 – El hook de Cursor no llega al daemon](#)
- [OL-1416 – Algunos agentes se quedaron sin daemon](#)
- [OL-1417 – Plugin del agente pendiente de su primera carga](#)
- [OL-1418 – Otro archivo ocupa la ruta del plugin](#)
- [OL-1419 – Versión del agente por debajo del mínimo admitido](#)
- [OL-1420 – Estado de un componente del agente desconocido](#)
- [OL-1421 – El plugin del agente no llega al daemon](#)
- [OL-1422 – El plugin del agente no se cargó](#)
- [OL-1423 – Relé del agente aún sin comprobar](#)
- [OL-1424 – Superficie del agente no cubierta por el relé](#)
- [OL-1425 – La herramienta de línea de comandos de un agente evita el relé](#)
- [OL-1426 – Aplicación de escritorio del agente aún sin registrar](#)
- [OL-1427 – El servicio de escritorio del agente no pudo iniciarse](#)
- [OL-1500 – Puerto ya en uso](#)
- [OL-1501 – El daemon ya está en marcha](#)
- [OL-1502 – El daemon no llegó a estar operativo](#)
- [OL-1503 – Nueva versión del cliente disponible](#)
- [OL-1504 – No se pudo aplicar la actualización](#)
- [OL-1505 – Compilado desde el código fuente, sin actualización automática](#)
- [OL-1506 – El daemon no respondió a la actualización](#)
- [OL-1507 – No se pudo detener el daemon](#)
- [OL-1510 – Una parte del daemon se reinició](#)
- [OL-1511 – Una parte del daemon sigue fallando](#)
- [OL-1512 – Daemon degradado](#)
- [OL-1513 – Nada reinicia el daemon](#)
- [OL-1600 – No se encontró ninguna credencial](#)
- [OL-1601 – Credencial caducada](#)
- [OL-1602 – Llaverio no disponible](#)
- [OL-1603 – El llaverio denegó el acceso](#)
- [OL-1604 – Falló el archivo de credencial cifrado](#)
- [OL-1605 – Se agotó el tiempo de inicio de sesión](#)
- [OL-1606 – Falló el inicio de sesión](#)

- [OL-1607 – No se pudo revocar la clave al cerrar sesión](#)
- [OL-1608 – Hay que elegir una organización](#)
- [OL-1609 – El selector de organización quedó sin respuesta](#)
- [OL-1610 – No se encontró la organización](#)
- [OL-1611 – El nombre de la organización es ambiguo](#)
- [OL-1612 – La clave de API pertenece a otra organización](#)
- [OL-1613 – No se pudo confirmar la organización](#)
- [OL-1614 – La máquina está inscrita en otra organización](#)
- [OL-1615 – No se aplicó el cambio de organización](#)
- [OL-1616 – Cambio rechazado: la clave viene del entorno](#)
- [OL-1617 – El código de inicio de sesión caducó](#)
- [OL-1618 – La cuenta no tiene organización](#)
- [OL-1619 – La organización no está disponible](#)
- [OL-1620 – Demasiados inicios de sesión en curso](#)
- [OL-1700 – Archivo de consentimiento de telemetría dañado](#)
- [OL-1701 – No se puede escribir el consentimiento de telemetría](#)
- [OL-1702 – Lote de telemetría no enviado](#)
- [OL-1703 – No se pudo iniciar la telemetría](#)
- [OL-1704 – Entrega de telemetría rechazada](#)
- [OL-1800 – Registro de reparación de doctor dañado](#)
- [OL-1801 – Nada que restaurar](#)
- [OL-1802 – No se puede escribir el archivo de diagnóstico](#)
- [OL-1803 – Archivo de diagnóstico incompleto](#)
- [OL-1900 – Clave de integridad de los hooks no disponible](#)
- [OL-1901 – Archivo de estado de los hooks dañado](#)
- [OL-1902 – No se puede escribir el archivo de estado de los hooks](#)
- [OL-1903 – Versión del marcador de hook no admitida](#)
- [OL-1904 – No se pudo procesar una entrada de hook](#)
- [OL-1905 – Marcador de hook antiguo actualizado](#)
- [OL-1906 – Hooks modificados reparados](#)
- [OL-1910 – No se puede sustituir el archivo de ajustes](#)
- [OL-1911 – No se puede resolver el enlace del archivo de ajustes](#)

- [OL-1950 – Supervisión no admitida en este sistema](#)
- [OL-1951 – No se pudo instalar el supervisor](#)
- [OL-1952 – No se pudo registrar el supervisor](#)
- [OL-1953 – El supervisor rechazó un comando](#)
- [OL-2000 – No se pudo iniciar el seguimiento de configuración](#)
- [OL-2001 – No se encontró la lista de fuentes de configuración](#)
- [OL-2002 – Lista de fuentes de configuración no válida](#)
- [OL-2003 – No se puede expandir una ruta seguida](#)
- [OL-2004 – No se puede seguir una ruta](#)
- [OL-2005 – No se pudo calcular la huella de un archivo de configuración](#)
- [OL-2006 – Nuevo análisis de configuración parcial](#)
- [OL-2007 – Detalles de inicio de sesión no disponibles](#)
- [OL-2008 – Seguimiento de configuración desactivado](#)
- [OL-9999 – Error interno inesperado](#)
- [OL-RELAY-CA – Certificado del relé de modelo no fiable](#)
- [OL-RELAY-CURSOR-UNPROVEN – Relé de Cursor aún sin comprobar](#)
- [OL-RELAY-CURSOR-UNSUPPORTED – Superficie de Cursor no cubierta por el relé](#)
- [OL-RELAY-DELIVERY – La ruta del agente hacia el relé no está lista](#)
- [OL-RELAY-DOWN – El relé de modelo no está en marcha](#)
- [OL-RELAY-FOREIGN – Otro proceso ocupa el puerto del relé](#)
- [OL-RELAY-IO – Error de entrada o salida del relé de modelo](#)
- [OL-RELAY-MISCONFIG – Ajuste del relé sustituido](#)
- [OL-RELAY-NEXTSTART – El relé se aplica en el próximo inicio del editor](#)
- [OL-RELAY-NOFIND – Ningún registro local con este identificador](#)
- [OL-RELAY-PORT – Puerto del relé de modelo ocupado](#)
- [OL-RELAY-PORTS – Ningún puerto de relé para un proveedor](#)
- [OL-RELAY-PREFLIGHT – El relé de modelo no llega al proveedor](#)
- [OL-RELAY-PROVIDER – La configuración del agente contiene otra entrada de proveedor](#)
- [OL-RELAY-SERVE – El relé de modelo se detuvo por un error](#)
- [OL-RELAY-SETTINGS – Los ajustes del agente contienen tu propio proxy](#)
- [OL-RELAY-STATEFILE – No se puede editar de forma segura el archivo de estado del agente](#)
- [OL-RELAY-TRUSTPROOF – Confianza del relé aún sin comprobar aquí](#)

- OL-RELAY-UNCOVERED – Proveedor no cubierto por el relé

Registro de cambios

Producto Todos los productos ▼ 49 entradas

Atom: [todo](#) · [cliente](#) · [plataforma](#)

13 de octubre de 2026 Cliente

Cliente 0.6.19

`openlatch init` y `openlatch doctor` indican si las llamadas al modelo de cada agente están listas para registrarse, y la instalación en Windows pide menos.

- `openlatch init` y `openlatch doctor` indican, para cada agente, si sus llamadas al modelo están listas para registrarse, y qué falta todavía.
- `openlatch init` pregunta antes de reiniciar un agente que ya estaba en marcha, y nombra los que te deja reiniciar a ti.
- Un Cursor que ya estaba abierto durante `openlatch init` queda cubierto en cuanto lo cierras y lo vuelves a abrir; recargar sus ventanas no basta.
- En Windows, la instalación ya no muestra un aviso del Firewall, y el diálogo de confianza del certificado va precedido de su huella digital para que puedas comprobarla.
- En Windows, correcciones de estabilidad para Cline, para una reinstalación después de `openlatch uninstall --purge`, y para los reinicios.
- Trabajo preparatorio para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.

13 de octubre de 2026 Cliente

Cliente 0.6.18

Elige tu organización al iniciar sesión y mueve una máquina a otra organización.

- Al iniciar sesión se pregunta qué organización usar cuando tu cuenta pertenece a varias; `--org` o `OPENLATCH_ORG` la nombra sin preguntar.
- El nuevo comando `openlatch system auth switch` mueve esta máquina a otra de tus organizaciones.

- En Windows, `openlatch doctor` informa de la herramienta de línea de comandos de un agente cuya instalación para toda la máquina evita el relé (OL-1425).
 - Trabajo preparatorio para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.
-

12 de octubre de 2026 Cliente

Ciente 0.6.17

Cursor vuelve a estar cubierto en cuanto se recargan sus ventanas.

- Un Cursor en espera vuelve a estar cubierto en cuanto se recargan sus ventanas.
 - El relé registra cada llamada de Cursor una sola vez y clasifica el tráfico de un agente de código que OpenLatch aún no admite.
-

12 de octubre de 2026 Cliente

Ciente 0.6.16

El cliente también conecta la aplicación de escritorio de Codex, sin herramienta de línea de comandos y sin un nuevo `openlatch init`.

- El cliente conecta la aplicación de escritorio de Codex en una máquina sin herramienta de línea de comandos de Codex, sin volver a ejecutar `openlatch init`.
 - `openlatch doctor` ya no informa de un falso problema para un agente de código que OpenLatch aún no admite.
-

12 de octubre de 2026 Plataforma

Una política en varias Autonomy Zones

Vincula una Agent Intent Policy a varias zonas, cada una con su propio modo.

11 de octubre de 2026 Cliente

Ciente 0.6.15

En Windows, las herramientas de línea de comandos de Cursor y Cline pasan por el relé sin configuración.

- En Windows, las herramientas de línea de comandos de Cursor y Cline envían su tráfico de modelo a través del relé sin configuración.

- En Windows, la comprobación del certificado del relé ya no lee un archivo que sigue abierto.
-

11 de octubre de 2026 Cliente

Cliente 0.6.14

Trabajo preparatorio para un agente de código que OpenLatch aún no admite.

- Trabajo preparatorio del relé para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.
-

10 de octubre de 2026 Cliente

Cliente 0.6.13

Trabajo preparatorio para un agente de código que OpenLatch aún no admite, y una corrección del relé en Windows.

- Trabajo preparatorio de hooks e inventario para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.
 - En Windows, el certificado del relé se comprueba en el contexto del propio usuario, y los procesos en segundo plano se inician sin abrir una ventana de consola.
-

10 de octubre de 2026 Cliente

Cliente 0.6.12

Un uso de modelos más exacto y una mejor lectura de PowerShell.

- `openlatch doctor --fix` informa del estado en que deja el equipo.
 - Se conserva el uso que un flujo de modelo interrumpido ya había informado.
 - Un modelo se marca como desconocido solo cuando su uso es estimado.
 - Una clave de API se distingue de un token de suscripción por su forma.
 - Un parámetro de PowerShell escrito como `-Param:value` se lee igual que `-Param value`.
-

9 de octubre de 2026 Cliente

Cliente 0.6.11

Los comandos de PowerShell se leen con más precisión.

- Una URL en un comando de PowerShell se lee ahora como una URL, no como una ruta de archivo.
-

8 de octubre de 2026 Cliente

Cliente 0.6.10

Correcciones para el registro y para la lectura de comandos de PowerShell.

- La comprobación de estado del daemon en reposo envía ahora la clave de API.
 - `openlatch init` guarda ahora un `OPENLATCH_API_URL` exportado en la configuración del cliente.
 - Una ruta de archivo se clasifica donde se resuelve, no como está escrita.
 - Los comandos de PowerShell se dividen en palabras según las propias reglas de PowerShell.
-

8 de octubre de 2026 Plataforma

Autonomy Zones en el árbol de tu organización

Activa una Autonomy Zone en cualquier unidad, y consulta qué agentes gobierna cada zona y por qué.

8 de octubre de 2026 Plataforma

Autonomy Zones y Agent Intent Policies

Define lo que cada agente puede hacer como Agent Intent Policies, y ejecuta cada una en modo Disabled, Monitor o Enforce.

8 de octubre de 2026 Plataforma

API REST pública

Automatiza las operaciones de la consola con claves de API de organización.

7 de octubre de 2026 Cliente

Cliente 0.6.9

Informes de configuración en directo, y correcciones para Cline, Cursor y macOS.

- Los cambios de configuración de un equipo se informan en el momento en que ocurren.
- Una sesión compañera de Claude Code se vincula a la sesión que la dirige.
- Las skills, reglas y servidores MCP que cargan Codex CLI, Cursor y Cline se inventarían tal como cada agente los carga realmente.

- Los informes de Cursor nombran el servidor MCP en cada llamada.
 - Actualizar Cline.app ya no interrumpe una conversación.
 - Una sesión de Claude Code iniciada desde un plan registra ese plan como su primer prompt.
 - En macOS, el cliente ya no solicita acceso a las carpetas Escritorio, Descargas y Documentos.
 - `openlatch init` ya no describe la telemetría de uso como anónima.
 - Las rutas nativas de Windows se clasifican correctamente en las lecturas y escrituras de archivos.
-

6 de octubre de 2026 Plataforma

Aislamiento de organizaciones aplicado por la base de datos

La propia base de datos mantiene separados los datos de cada organización.

6 de octubre de 2026 Cliente

Cliente 0.6.8

Correcciones del paquete de políticas y del relé.

- Un paquete de políticas en caché se revalida correctamente cuando la plataforma lo informa sin cambios.
 - Ya no se registra el uso de sesiones que ningún hook registró.
-

5 de octubre de 2026 Cliente

Cliente 0.6.7

Una evaluación que tiene en cuenta las rutas, y correcciones para Cline.

- Los efectos de una acción se relacionan con las rutas a las que apunta.
 - El plugin de Cline se aplica a todas las herramientas, no solo al shell.
 - `openlatch init` aguarda la primera entrega del plugin de Cline en lugar de fallar.
 - Cada acción se decide una sola vez, no otra vez después de ejecutarse.
 - Los hilos secundarios de Codex CLI se archivan bajo su sesión principal.
 - El uso se asigna a la instalación correcta.
-

5 de octubre de 2026 Plataforma

Políticas que leen lo que hace una acción

Las políticas de intención reconocen lo que hace un comando, como leer secretos o llegar a un host de red, no cómo está escrito.

2 de octubre de 2026 Cliente

Cliente 0.6.6

Las llamadas de modelo del IDE de Cursor pasan por el relé.

- Las llamadas de modelo del IDE de Cursor se enrutan a través del relé.
 - Un `openlatch init` nuevo ya no falla con componentes que aún se están iniciando.
 - `openlatch doctor` muestra una línea de estado por agente y reconoce el hub de Cline.app.
 - Una sustitución de procesos del shell ya no se lee como destino de una redirección.
-

1 de octubre de 2026 Cliente

Cliente 0.6.5

Un vocabulario de herramientas común, y veredictos Ask entregados según cada agente.

- Las herramientas de cada agente se asignan a un vocabulario común; una herramienta sin asignación se registra igualmente con su propio nombre.
 - La evaluación de políticas lee un único catálogo de clasificación del paquete de políticas.
 - Los veredictos Ask llegan a cada agente en la forma que admite, con un veredicto definido de antemano cuando un agente no tiene un ask propio.
-

1 de octubre de 2026 Cliente

Cliente 0.6.4

Compatibilidad con Cursor.

- Las políticas deciden sobre las acciones de Cursor, y el veredicto llega a Cursor en su propio formato.
 - El tráfico de modelo de Cursor pasa por el relé y se registra como uso, nunca como contenido.
 - La conexión de Cursor se comprueba, se repara y se inventaría como la de cualquier otro agente.
 - `openlatch uninstall` nombra los agentes que una desinstalación parcial deja conectados.
-

30 de septiembre de 2026 Cliente

Ciente 0.6.3

El plugin de Cline llega a Cline.app y a VS Code.

- El plugin de Cline se entrega a Cline.app y a VS Code.
 - El cliente comprueba que el plugin se cargó en cada host.
-

29 de septiembre de 2026 Plataforma

Licencia y consumo en la consola

Compara aquello para lo que tu organización tiene licencia con lo que consume, en una sola página.

29 de septiembre de 2026 Plataforma

Propiedad de los agentes

Cada agente se resuelve a un propietario de tu directorio, y su función de negocio puede guiar una política.

29 de septiembre de 2026 Plataforma

Agentes reconocibles

Los agentes llevan un nombre legible y muestran las sesiones y proyectos que agrupan.

29 de septiembre de 2026 Plataforma

Grafo de ejecución

Consulta todo lo que hizo un agente como un grafo, fíltralo, y crea una regla a partir de un comando.

29 de septiembre de 2026 Plataforma

Economía de los agentes

Consulta el gasto, la previsión y las sesiones de cada agente, con oportunidades de ahorro ordenadas.

29 de septiembre de 2026 Cliente

Ciente 0.6.2

Informes de Cline, autorreparación y una purga completa.

- Las sesiones, prompts y llamadas a herramientas de Cline se envían a la plataforma.

- El cableado del relé, la confianza de los hooks y los archivos del plugin de Cline se comprueban contra manipulaciones y se reparan.
 - El cliente informa cuando un paquete de políticas está pendiente de entrega.
 - `openlatch uninstall` envía un evento de desinstalación, y `openlatch uninstall --purge` elimina todo el estado del cliente.
-

29 de septiembre de 2026 Cliente

Cliente 0.6.1

Cline a través del relé, y gestión del certificado del relé.

- El tráfico de modelo de Cline se captura a través del relé, incluida la aplicación de escritorio de Cline.
 - El relé demuestra que su certificado es de confianza antes de que un agente se dirija a él.
 - El certificado del relé se puede consultar, rotar, liberar y eliminar.
 - Solo los agentes detectados aparecen en los datos del equipo.
 - Los hooks de Codex CLI son de confianza en Windows.
-

29 de septiembre de 2026 Plataforma

Políticas que siguen decidiendo sin conexión

Cada cliente guarda localmente su paquete de políticas, así que la aplicación continúa con la red desconectada.

24 de septiembre de 2026 Cliente

Cliente 0.6.0

Un `openlatch update` paso a paso, con un cambio incompatible en su salida JSON.

- Cambio incompatible: `openlatch update --check --json` ya no devuelve `min_supported_client.current`, `latest` y `severity` no cambian.
 - `openlatch update` muestra su progreso paso a paso, y el daemon vuelve a iniciarse después.
 - El inventario de configuración informa de los hooks, reglas y flujos de trabajo de Cline.
-

23 de septiembre de 2026 Cliente

Ciente 0.5.8

Captura de prompts de Cline.

- Los prompts se capturan en la CLI de Cline y en la aplicación de escritorio de Cline.
 - Ya no se informan los tipos de agente que no están conectados.
-

23 de septiembre de 2026 Ciente

Ciente 0.5.7

Corrección de la salida del terminal en Windows.

- Los colores y el progreso se muestran correctamente en Windows en todos los comandos, no solo en `openlatch init`.
-

23 de septiembre de 2026 Ciente

Ciente 0.5.6

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

22 de septiembre de 2026 Ciente

Ciente 0.5.4

Correcciones de la supervisión del daemon y del envío de telemetría.

- En Windows, la tarea programada que mantiene el daemon en marcha conserva su marcador de versión.
 - El primer lote de telemetría del daemon se envía por HTTPS.
-

22 de septiembre de 2026 Ciente

Ciente 0.5.3

La telemetría pasa por la plataforma, y los binarios de Linux arrancan en hosts más antiguos.

- La telemetría y los informes de fallos pasan por el punto de ingesta de la propia plataforma.
 - Los binarios de Linux arrancan en hosts mínimos y más antiguos.
-

21 de septiembre de 2026 Cliente

Ciente 0.5.2

Correcciones del arranque y del envío de eventos.

- En Windows, `openlatch` se ejecuta como un programa de consola, y sus binarios arrancan en un equipo recién instalado.
 - Se envían los eventos que seguían en cola al apagar.
 - El daemon detecta un nuevo inicio de sesión sin reiniciarse.
 - El envío de eventos se recupera tras cada reconexión con la plataforma.
-

20 de septiembre de 2026 Cliente

Ciente 0.5.1

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

8 de septiembre de 2026 Cliente

Ciente 0.4.0

Los comandos de los subsistemas pasan a `openlatch system`.

- Cambio incompatible: `openlatch auth, hooks, model-relay, proxy, supervision, telemetry` e `inventory` pasan a ser `openlatch system <name>`, sin alias.
 - Cambio incompatible: la sección de configuración `[boundary]` pasa a ser `[model_relay]`, las variables `OPENLATCH_BOUNDARY_*` pasan a ser `OPENLATCH_MODEL_RELAY_*`, y los códigos de error `OL-BND-*` pasan a ser `OL-RELAY-*`.
 - `openlatch doctor` informa del estado en directo de cada hook.
 - Los diagnósticos de proxy informan del proxy que el cliente usa realmente.
-

6 de septiembre de 2026 Cliente

Ciente 0.3.3

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

28 de agosto de 2026 Cliente

Cliente 0.2.2

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

19 de agosto de 2026 Cliente

Cliente 0.1.18

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

23 de abril de 2026 Cliente

Cliente 0.1.14

Corrección de la dirección de API predeterminada.

- La dirección de API predeterminada ya no repite `/api`.
-

23 de abril de 2026 Cliente

Cliente 0.1.13

El paquete npm se resuelve con `npx @openlatch/client init`.

- El paquete npm incorpora un punto de entrada `client`, de modo que `npx @openlatch/client init` se resuelve.
 - La ayuda ya no muestra un banner.
-

23 de abril de 2026 Cliente

Cliente 0.1.12

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

23 de abril de 2026 Cliente

Cliente 0.1.8

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

23 de abril de 2026 Cliente

Cliente 0.1.7

El paquete npm incluye el punto de entrada del hook.

- El paquete npm instala el punto de entrada `openLatch-hook`.
-

23 de abril de 2026 Cliente

Cliente 0.1.6

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.
-

OL-1002 — Evento demasiado grande

OL-1002 · ERR_EVENT_T00_LARGE

Qué ha pasado

El cuerpo del evento supera el límite de tamaño de 1 MB.

Qué hacer

Ejecuta `openlatch doctor` para comprobar el cliente. Si el error se repite, genera un archivo de diagnóstico con `openlatch doctor --rescue` y envíalo al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1003 — Evento duplicado ignorado

OL-1003 · ERR_EVENT_DEDUPED

Qué ha pasado

El evento se deduplicó dentro de la ventana de TTL (informativo).

Qué hacer

No hace falta hacer nada. Si ocurre a menudo, ejecuta `openlatch doctor` para comprobar que los hooks de cada agente están instalados una sola vez.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1100 — La expresión personalizada no es válida

OL-1100 · ERR_INVALID_REGEX

Qué ha pasado

Un patrón de expresión regular personalizado de `config.toml` no se pudo compilar.

Qué hacer

Corrige la expresión en `~/.openlatch/config.toml` y ejecuta `openlatch restart` para cargarla.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1200 — Plataforma inaccesible

OL-1200 · ERR_CLOUD_UNREACHABLE

Qué ha pasado

El endpoint de la API de Cloud no está accesible (error de red o fallo de DNS).

Qué hacer

Ejecuta `openlatch system proxy test` para comprobar la ruta entre esta máquina y la plataforma.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1201 — La plataforma rechazó la credencial

OL-1201 · ERR_CLOUD_AUTH_FAILED

Qué ha pasado

Cloud devolvió 401/403: la clave de API no es válida o se revocó.

Qué hacer

Vuelve a iniciar sesión con `openlatch system auth login`, o define una clave válida en `OPENLATCH_API_KEY` y ejecuta `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1202 — Límite de frecuencia de la plataforma alcanzado

OL-1202 · ERR_CLOUD_RATE_LIMITED

Qué ha pasado

Cloud devolvió 429: se superó el límite de solicitudes; respeta el encabezado Retry-After.

Qué hacer

No hace falta hacer nada si se resuelve solo. Si persiste, ejecuta `open\latch status` y comparte el resultado con el soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1204 — No se pudo escribir en el outbox

OL-1204 · ERR_OUTBOX_WRITE_FAILED

Qué ha pasado

Fallo de escritura en la outbox duradera (disco lleno, permiso denegado, error de E/S). El daemon sigue atendiendo eventos —la outbox es una ayuda para la recuperación, no un mecanismo de corrección—, pero el evento se pierde para la reproducción hacia Cloud.

Qué hacer

Libera espacio en disco o corrige los permisos de `~/openlatch`, y ejecuta `openlatch doctor` para confirmar que se puede escribir en el outbox.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1205 — Outbox enviado solo en parte

OL-1205 · ERR_OUTBOX_DRAIN_PARTIAL

Qué ha pasado

El vaciado de la outbox terminó con algunas entradas todavía pendientes (fallo transitorio de Cloud a mitad de la reproducción, o error de E/S al leer el spool). Las entradas restantes se conservan en disco y se reintentan en la siguiente señal de vaciado.

Qué hacer

No hace falta hacer nada si el próximo intento funciona. Si siguen quedando entradas pendientes, ejecuta `openlatch doctor`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1206 — Canal hacia la plataforma saturado

OL-1206 · ERR_CLOUD_CHANNEL_EMERGENCY

Qué ha pasado

El canal de reenvío a Cloud sufre una congestión sostenida: se activó el modo de emergencia. Los eventos de hook en vivo se descartaban más rápido de lo que el worker podía vaciar la cola; la reproducción está en pausa y la cola de respaldo puede desalojarse para recuperar margen. Se desactiva automáticamente cuando los descartes cesan durante dos ciclos de salud consecutivos.

Qué hacer

Ejecuta `openlatch status` para seguir el número de eventos, y `openlatch system proxy test` si la conexión con la plataforma es lenta.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1207 — Entrada del outbox apartada

OL-1207 · ERR_OUTBOX_QUARANTINED

Qué ha pasado

Una entrada de la outbox falló repetidamente durante la vida actual del daemon y se puso en cuarentena para que la cola pueda vaciarse más allá de ella. La entrada se elimina de la outbox sin llegar nunca a Cloud; solo se emite cuando el número de intentos por identificador supera `OUTBOX_MAX_ATTEMPTS`.

Qué hacer

Ejecuta `openlatch doctor --rescue` y envía el archivo de diagnóstico al soporte para que se examine la entrada.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1208 — Evento retenido caducado

OL-1208 · ERR_OUTBOX_EXPIRED

Qué ha pasado

Un evento de compatibilidad retenido alcanzó su caducidad duradera explícita antes de poder entregarse. Es una caducidad definitiva, no un problema de autenticación ni de transporte.

Qué hacer

Ejecuta `openlatch doctor` para comprobar la conexión con la plataforma, para que los eventos se entreguen antes de caducar.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1210 — No se pudo obtener el paquete de políticas

OL-1210 · ERR_BUNDLE_FETCH_FAILED

Qué ha pasado

Falló una consulta del policy bundle (error de red, 5xx o 401/403). El bundle activado anteriormente se conserva y sigue aplicando las reglas: fail-static, nunca fail-open. Nivel de advertencia para los fallos transitorios y nivel de error para los fallos de autenticación.

Qué hacer

Ejecuta `openlatch doctor` para encontrar la causa y luego `openlatch system aip refresh` para obtener de nuevo el paquete.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1211 — El paquete de políticas no superó la verificación

OL-1211 · ERR_BUNDLE_REJECTED

Qué ha pasado

Un bundle descargado o guardado en disco no superó la verificación y NO se activó: la huella no coincide con el ETag, un `organization_id` que no es el nuestro, una respuesta 200 sin encabezado ETag (la huella no está dentro del cuerpo, así que un cuerpo no verificable nunca debe activarse), o un `bundle.json` en caché cuyos bytes ya no coinciden con `bundle.meta.json`. El bundle anterior sigue residente; el validador almacenado deliberadamente NO avanza, de modo que la siguiente consulta vuelve a pedir el bundle y reintenta la activación.

Qué hacer

Ejecuta `openlatch system aip refresh` para obtener una copia nueva. Si el error se repite, ejecuta `openlatch doctor --rescue` y envía el archivo al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1212 — El paquete de políticas no se puede usar

OL-1212 · ERR_BUNDLE_INVALID

Qué ha pasado

No se pudo leer un paquete de políticas: está mal formado, usa una versión de formato que este cliente no conoce o lleva una firma que este cliente no puede comprobar.

Qué hacer

Actualiza el cliente con `openlatch update` y ejecuta `openlatch system aip refresh`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1213 — Actualización del paquete de políticas atrasada

OL-1213 · ERR_BUNDLE_STALE

Qué ha pasado

No hubo ninguna consulta correcta dentro de `stale_warn_after_secs`. Solo una advertencia: el bundle residente sigue aplicando las reglas. Se mide desde la última consulta correcta (conectividad), no desde `built_at` (antigüedad de la política), y una respuesta 304 cuenta como consulta correcta.

Qué hacer

Comprueba la conexión de la máquina con la plataforma mediante `openlatch doctor`. Cuando se restablezca, ejecuta `openlatch system aip refresh` para obtener el último paquete de inmediato.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1214 — Una regla del paquete quedó apartada

OL-1214 · ERR_RULE_SKIPPED

Qué ha pasado

Una regla dentro de un bundle por lo demás válido se trató individualmente en el momento de la proyección —se descartó o se mantuvo en un modo más débil— y el resto del bundle se activó con normalidad.

Deliberadamente NO es OL-1212, que significa *el bundle se rechazó y se conservó el anterior*; reutilizarlo invertiría su significado para quien tenga alertas sobre él.

Qué hacer

Actualiza el cliente con `openlatch update` para que admita todas las reglas del paquete, y ejecuta `openlatch system aip refresh`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1215 — La política local está desactivada

OL-1215 · ERR_POLICY_DISABLED

Qué ha pasado

El plano de políticas está desactivado en la configuración, así que no se evalúa ninguna regla y se permite toda acción.

Qué hacer

Quita `enabled = false` de la sección `[policy]` de `~/.openlatch/config.toml` (o elimina `OPENLATCH_POLICY_ENABLED`) y ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1216 — La nueva versión no puede leer la caché de políticas

OL-1216 · ERR_READER_RETIREMENT_BLOCKED

Qué ha pasado

Un ejecutable candidato no puede cargar la caché de políticas residente exacta y, por tanto, no puede sustituir al evaluador que sí puede.

Qué hacer

Ejecuta `openlatch system aip refresh` y vuelve a intentar la actualización con `openlatch update`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1220 — Sin ruta hacia la plataforma

OL-1220 · ERR_EGRESS_UNREACHABLE

Qué ha pasado

No se pudo llegar a la plataforma en absoluto: no hay ningún proxy configurado y la conexión directa a la plataforma falló a nivel de TCP/DNS. Distinto de OL-1221, más abajo: ese código se produce cuando hay un **proxy** en el camino y no está accesible, mientras que este significa que el tráfico saliente nunca tuvo un proxy al que culpar. Revisa la resolución DNS saliente y las reglas de firewall del host que ejecuta el daemon.

Qué hacer

Comprueba la resolución DNS y las reglas de cortafuegos de esta máquina, y ejecuta `openlatch system proxy test` para confirmar la ruta.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1221 — Proxy inaccesible

OL-1221 · ERR_PROXY_UNREACHABLE

Qué ha pasado

Hay un proxy configurado, pero el propio proxy no respondió: la conexión TCP al host del proxy falló, o un `CONNECT HTTPS` a través de él expiró o se rechazó. Nunca se llega al destino porque el cliente ni siquiera puede establecer el túnel hacia él. Comprueba que el host y el puerto del proxy sean correctos y estén accesibles desde esta máquina antes de sospechar del destino.

Qué hacer

Comprueba el host y el puerto del proxy con `openlatch system proxy status`, y ejecuta `openlatch system proxy test`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1222 — El proxy rechazó la credencial

OL-1222 · ERR_PROXY_AUTH_FAILED

Qué ha pasado

El proxy respondió 407 Proxy Authentication Required: o se rechazaron las credenciales que ofreció el cliente, o no tenía ninguna que ofrecer. Distinto de OL-1201/ERR_CLOUD_AUTH_FAILED: ese es la *plataforma* rechazando nuestra clave de API; este es el control propio del proxy, que salta antes de que la solicitud llegue a la plataforma. Una credencial que ayer era válida y un proxy corporativo que cambió su contraseña se ven idénticos en la red, así que el operador debe volver a confirmar específicamente la credencial del proxy, no la clave de API de la plataforma.

Qué hacer

Define la credencial actual del proxy con `openlatch system proxy set` y ejecuta `openlatch system proxy test`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1223 — Esquema de proxy no admitido

OL-1223 · ERR_PROXY_SCHEME_UNSUPPORTED

Qué ha pasado

La URL de proxy configurada indica un esquema que esta versión no sabe usar: cualquiera fuera de `http/https/socks5/socks5h`, o `negotiate` solicitado en una versión compilada sin la funcionalidad Kerberos. Falla de inmediato al resolver el proxy en lugar de recurrir en silencio a una conexión simple: una degradación silenciosa haría que el tráfico rodeara el mismo proxy que configuró el operador. Corrige el esquema en la URL del proxy o recompila con la funcionalidad que ese esquema requiere.

Qué hacer

Define una URL de proxy que empiece por `http`, `https`, `socks5` o `socks5h` con `openlatch system proxy set`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1224 — Certificado TLS no fiable

OL-1224 · ERR_EGRESS_TLS_FAILED

Qué ha pasado

El handshake TLS con el destino no superó la verificación del certificado. En una red empresarial, la causa más común es un proxy o firewall con inspección TLS que vuelve a firmar el tráfico saliente con una CA de interceptación que no está en el almacén de confianza del sistema y no se proporcionó mediante `ca_bundle`; desde el punto de vista del cliente, eso es idéntico a una interceptación real, y precisamente por eso rechaza el handshake en lugar de adivinar. La solución casi siempre es apuntar `ca_bundle` a la raíz de interceptación, nunca desactivar la verificación.

Qué hacer

Apunta `OPENLATCH_CA_BUNDLE` (o `ca_bundle` en la sección `[proxy]`) al certificado raíz de tu red y ejecuta `openlatch system proxy test`. Nunca desactives la verificación.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1225 — Configuración automática del proxy no disponible

OL-1225 · ERR_PAC_UNAVAILABLE

Qué ha pasado

Hay una fuente PAC (Proxy Auto-Config) o WPAD configurada, pero el sistema no pudo evaluarla: la URL PAC no se resolvió o no sirvió un script, o el motor PAC de la plataforma no está disponible en este sistema. El tráfico saliente recurre al proxy estático o al comportamiento directo que la configuración declare bajo la entrada PAC, así que este código es informativo salvo que ese respaldo también falle.

Qué hacer

Ejecuta `openlatch system proxy discover` para encontrar una ruta que funcione, o define una a mano con `openlatch system proxy set`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1226 — Configuración del proxy no válida

OL-1226 · ERR_PROXY_CONFIG_INVALID

Qué ha pasado

La propia configuración del proxy está mal formada: una URL de proxy que no se puede analizar, un esquema al que le falta un campo de credencial obligatorio, o `allow_direct` y la configuración del proxy que se contradicen. A diferencia de todos los demás códigos de este bloque —cada uno describe un estado de red en tiempo de ejecución que nunca bloquea el arranque—, este **Sí** hace fallar el arranque: una configuración que el cliente ni siquiera puede analizar es un error de redacción, no un estado de red transitorio, y dejar que el daemon arranque con ella solo trasladaría el fallo a la primera solicitud.

Qué hacer

Corrige el proxy con `openlatch system proxy set`, o elimínalo con `openlatch system proxy clear`, y ejecuta `openlatch start`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1227 — Conexión directa no permitida

OL-1227 · ERR_DIRECT_FORBIDDEN

Qué ha pasado

`allow_direct = false` y no se encontró ningún proxy utilizable para este destino. El tráfico saliente rechaza la solicitud en lugar de recurrir en silencio a una conexión directa: en una red cuya política exige que todo el tráfico saliente pase por un proxy, un respaldo directo silencioso es una vía de exfiltración, no una comodidad. Corrige la configuración del proxy, o establece `allow_direct = true` si el tráfico saliente directo desde este host es realmente aceptable.

Qué hacer

Arregla el proxy con `openlatch system proxy discover` o `openlatch system proxy set`. Define `allow_direct = true` solo si las conexiones directas desde esta máquina son aceptables.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1228 — El proxy almacena en búfer las respuestas en streaming

OL-1228 · ERR_STREAM_BUFFERED

Qué ha pasado

El proxy almacena en búfer una respuesta en streaming en lugar de pasar los bytes a medida que llegan: el túnel `CONNECT` en sí funciona, pero un flujo de Server-Sent Events se acumula en el búfer interno del proxy y llega de golpe en lugar de transmitirse. Es un diagnóstico del comportamiento del proxy, no un fallo de conectividad —la solicitud sigue teniendo éxito—, así que se muestra como advertencia en lugar de hacer fallar la llamada. Busca en el proxy un ajuste de búfer de respuestas o de inspección de contenido para el endpoint de streaming.

Qué hacer

Pide a tu equipo de red que desactive el almacenamiento en búfer o la inspección de contenido para la plataforma en el proxy, y ejecuta `openlatch system proxy test`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1229 — Sin ticket Kerberos para el proxy

OL-1229 · ERR_NEGOTIATE_NO_TICKET

Qué ha pasado

Se eligió `negotiate` (Kerberos/SPNEGO) para la autenticación del proxy, pero no se pudo obtener ningún ticket utilizable: no hay caché de credenciales de Kerberos, el TGT caducó o el SPN del proxy no se resuelve desde este host. Distinto de OL-1222/ERR_PROXY_AUTH_FAILED: al proxy ni siquiera se le ofreció un token para aceptar o rechazar. La solución está en el host: ejecuta `kinit`, o confirma que la máquina está unida al dominio para que Windows pueda obtener un ticket automáticamente.

Qué hacer

Ejecuta `kinit`, o comprueba que la máquina está unida al dominio, y ejecuta `openlatch system proxy test`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1240 — Fila de entrada de evaluación mal formada

OL-1240 · ERR_EVAL_ROW_MALFORMED

Qué ha pasado

Una trama row que no se deserializó como fila de solicitud: un event que no es un objeto, un now_ms que no es un entero, un state_in que no es un archivo de registros. ****Una fila, nunca el flujo****: la trama recibe una respuesta de error y la ejecución continúa, de modo que una reproducción informa «se evaluaron todos los eventos menos cuatro» en lugar de detenerse a mitad de camino.

Qué hacer

Corrige la fila indicada en la salida y vuelve a ejecutar la evaluación.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1241 — Paquete de evaluación rechazado

OL-1241 · ERR_EVAL_BUNDLE_REJECTED

Qué ha pasado

Falló la etapa 1 del análisis del bundle, así que **no se cargó nada**: el sobre no se deserializó, o su `client_floor` supera lo que implementa este motor. Se escribe como una trama `bundle_error`, un resultado distinto de un `bundle_ack` con un `skipped` no vacío: ese significa que el bundle Sí está cargado y que un solo artefacto no lo está.

Qué hacer

Actualiza el cliente con `openlatch update` y vuelve a ejecutar la evaluación.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1242 — Ningún paquete con el que evaluar

OL-1242 · ERR_EVAL_NO_BUNDLE

Qué ha pasado

Llegó una row sin un bundle con el que evaluarla: no se ha aceptado ninguna trama de encabezado y la fila no lleva su propio bundle en línea. Se responde fila por fila, porque la siguiente trama de encabezado bien puede corregirlo.

Qué hacer

Envía el paquete de políticas antes de las filas, o incluye un paquete con la fila.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1243 — El estado de sesión no coincide con el paquete

OL-1243 · ERR_EVAL_STATE_MISMATCH

Qué ha pasado

El `state_in` de la fila no concuerda con el `state_layout` del bundle. ****Mal formado, nunca relleno****: dos implementaciones no pueden ponerse de acuerdo sobre qué contador es cuál si una de ellas inventa en silencio una posición.

Qué hacer

Envía el estado de sesión que el cliente devolvió para la fila anterior de la misma sesión.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1244 — Falló la evaluación de una fila

OL-1244 · ERR_EVAL_PANICKED

Qué ha pasado

El motor entró en pánico al evaluar una fila.

Qué hacer

Ejecuta `openlatch doctor --rescue` y envía el archivo de diagnóstico al soporte junto con la fila que falló.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1250 — Recepción pausada por motivos de licencia

OL-1250 · ERR_CLOUD_LICENSE_BLOCKED

Qué ha pasado

La plataforma rechazó la ingesta por motivos de **licencia**: un 402, o un 503 cuyo cuerpo indica un código de licencia de sitio.

Qué hacer

Sigue el enlace de licencia que muestra `openlatch status`, o contacta con tu representante de OpenLatch si no aparece ninguno.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1300 — Valor de configuración no válido

OL-1300 · ERR_INVALID_CONFIG

Qué ha pasado

El archivo de configuración contiene un valor no válido.

Qué hacer

Ejecuta `openlatch doctor` para encontrar el valor, corrígelo en `~/.openlatch/config.toml` y ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1301 — Falta un campo de configuración obligatorio

OL-1301 · ERR_MISSING_CONFIG_FIELD

Qué ha pasado

Falta un campo de configuración obligatorio y no tiene valor predeterminado.

Qué hacer

Ejecuta `openlatch init` para volver a escribir una configuración completa.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1302 — Configuración cambiada pero no aplicada

OL-1302 · ERR_CONFIG_NOT_APPLIED

Qué ha pasado

El archivo en disco cambió desde que el daemon en ejecución lo leyó, así que el proceso sigue sirviendo los valores anteriores. El archivo no tiene ningún problema —`openlatch system proxy set` y los comandos similares lo escriben correctamente y lo indican— y el daemon tampoco. Simplemente están desincronizados, y solo un reinicio cierra la diferencia.

Qué hacer

Ejecuta `openlatch restart` para aplicar la nueva configuración.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1400 — No se encontró ningún agente compatible

OL-1400 · ERR_HOOK_AGENT_NOT_FOUND

Qué ha pasado

No se detectó ningún agente de IA compatible en esta máquina.

Qué hacer

Instala un agente compatible y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1401 — No se pueden escribir los ajustes del agente

OL-1401 · ERR_HOOK_WRITE_FAILED

Qué ha pasado

No se puede leer ni escribir el settings.json del agente (permisos, error de E/S).

Qué hacer

Corrige los permisos del archivo de ajustes del agente y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1402 — No se puede analizar el archivo de ajustes del agente

OL-1402 · ERR_HOOK_MALFORMED_JSONC

Qué ha pasado

El archivo `settings.json` contiene JSONC mal formado que no se puede analizar.

Qué hacer

Corrige el error de sintaxis en el archivo `settings.json` del agente y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1403 — Otros hooks en los ajustes del agente

OL-1403 · ERR_HOOK_CONFLICT

Qué ha pasado

Se detectaron hooks existentes ajenos a OpenLatch en settings.json (advertencia, no bloqueante).

Qué hacer

No hace falta hacer nada salvo que esos hooks sean inesperados. Ejecuta `openlatch system hooks status` para ver todos los hooks instalados.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1404 — No se puede instalar el binario del hook

OL-1404 · ERR_HOOK_BINARY_UNRESOLVABLE

Qué ha pasado

No se pudo colocar ningún binario `openlatch-hook` en `<ol_dir>/bin/`, así que cualquier comando de hook escrito ahora se resolvería como un nombre sin ruta y terminaría con el código de salida 127 en cada llamada a herramienta del agente. Fatal para `init` —una instalación que no puede resolver su propio binario de hook no es una instalación correcta— y lo informa (sin aplicarlo) `doctor --fix`.

Qué hacer

Comprueba los permisos y el espacio libre en `~/ .openlatch` y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1405 — Hooks instalados pero sin ejecutarse

OL-1405 · ERR_HOOK_NOT_ARMED

Qué ha pasado

Los hooks del agente están instalados y son correctos en disco, y aun así el agente nunca los ejecutará, así que no se aplica nada. Tres formas de llegar aquí, un código y tres soluciones: nuestro binario de hook no se resuelve; al agente no se le indicó que confíe en el hook (Codex marca un comando de hook nuevo o modificado para revisión y lo omite hasta que el desarrollador ejecuta `/hooks`); o un ajuste administrativo suprime en bloque los hooks no gestionados.

Qué hacer

Ejecuta `openlatch doctor`: indica cuál de las tres causas encontró y el paso que la corrige.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1406 — No se puede analizar la configuración TOML del agente

OL-1406 · ERR_HOOK_MALFORMED_TOML

Qué ha pasado

Hay un archivo de configuración de agente en TOML —el `config.toml` de Codex CLI— que no es TOML válido, así que la tabla de proveedor del model relay no se puede escribir ni revertir sin destruir lo que ya contiene.

Qué hacer

Corrige el error de sintaxis en el `config.toml` del agente y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1407 — El agente seleccionado no admite hooks

OL-1407 · ERR_HOOK_AGENT_NOT_INSTALLABLE

Qué ha pasado

El agente está instalado en esta máquina y no se puede escribir en absoluto en su superficie de hooks, así que `--agent` no seleccionó nada sobre lo que este comando pueda actuar.

Qué hacer

Vuelve a ejecutar el comando sin `--agent`, o indica un agente que esta versión admita.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1408 — Falta un componente esperado de Cline

OL-1408 · ERR_CLINE_SURFACE_ABSENT

Qué ha pasado

Una superficie de Cline que esta versión esperaba y no encontró: la raíz del almacenamiento, el directorio Hooks/, plugins/, el registro MCP o una extensión del linaje de Cline en alguna raíz de editor conocida.

Qué hacer

Ejecuta `openlatch doctor` para ver qué parte falta, instala o actualiza Cline y ejecuta `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1409 — Estado de un componente de Cline desconocido

OL-1409 · ERR_CLINE_SURFACE_UNDETERMINED

Qué ha pasado

Una superficie de Cline cuyo estado no se pudo determinar.

Qué hacer

Ejecuta `openlatch doctor` para ver qué ruta falló, y corrige sus permisos.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1410 — Las acciones de Cline no se pueden rechazar

OL-1410 · ERR_CLINE_NOT_ENFORCING

Qué ha pasado

Cline está instalado, sus diez scripts de hook están en disco y capturando, y **nada** de lo que observan se puede rechazar**.**

Qué hacer

Ejecuta `openlatch doctor` para saber por qué no está cargado el plugin, y sigue la solución que indica.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1411 — Llamadas al modelo de Cline.app no capturadas

OL-1411 · ERR_CLINE_APP_NOT_CAPTURED

Qué ha pasado

El tráfico de modelo de Cline.app no pasa por el relay: su hub se ejecuta sin la captura de esta instalación (se inició antes que ella o desde una versión anterior de Cline.app), el puerto del hub lo ocupa otra cosa, el hub se gestiona en una plataforma donde eso no está probado (D-47), la plataforma no tiene una ruta de instalación establecida y la aplicación no se puede detectar (A-8), o 06 §0 demostró que la aplicación no se puede capturar (§F). Una sola condición, «Cline.app no está capturado»; el motivo aparece en la línea de detalle.

Qué hacer

Cierra todas las conversaciones de Cline y sal de Cline.app; luego ábrela de nuevo para que su hub se reinicie a través del relé.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1412 — Plugin de OpenLatch no cargado en Cline

OL-1412 · ERR_CLINE_PLUGIN_NOT_LOADED

Qué ha pasado

Un host de Cline (el hub de Cline.app, la extensión de VS Code) no puede cargar o no carga el plugin de OpenLatch, así que allí no se puede entregar un veredicto y el host queda solo en Monitor a través de los hooks de archivo. Una sola condición; el motivo es el detalle: `host-too-old`, `delivery-off`, `probe-failed`, `pending-restart`, `runtime-missing`, `windows-launcher-unproven`, `not-managed`, `customer-override`, `not-detected-on-platform`, `windows-launcher-missing`. Desactivado nunca cuenta como correcto (`cli-output-contract`).

Qué hacer

Ejecuta `openlatch doctor` para conocer el motivo. Si la entrega del plugin está desactivada, `defineplugin_delivery = true` en la sección `[cline]` de `~/.openlatch/config.toml` y ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1413 — Formato desconocido del archivo de hooks de Cursor

OL-1413 · ERR_CURSOR_HOOK_FILE_MALFORMED

Qué ha pasado

El `hooks.json` de Cursor no es un archivo en el que esta versión pueda escribir con seguridad: no se puede analizar, o su `version` raíz está presente y no es el entero 1 que documenta Cursor. La instalación se niega en lugar de escribir en un formato que no conoce: una suposición ahí podría impedir que se cargue todo el archivo, incluidos los hooks del propio desarrollador. La solución es corregir o mover el archivo y volver a ejecutar `openlatch init`.

Qué hacer

Corrige el archivo o muévelo a otro lugar, y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1414 — No se puede confirmar la instalación de Cursor

OL-1414 · ERR_CURSOR_SURFACE_UNDETERMINED

Qué ha pasado

Una superficie de Cursor (el IDE o la CLI) tiene en este host una ubicación de instalación que no se puede confirmar como Cursor: la aplicación está presente, pero su `resources/app/package.json` falta, no se puede leer o no nombra ningún producto de Cursor, o el directorio de versiones de la CLI existe pero no se puede leer. `doctor` lo informa como degradado, nombrando la superficie y lo que no pudo leer. La versión se lee solo de archivos: `doctor` nunca ejecuta Cursor, cuyo binario de macOS abre la interfaz gráfica.

Qué hacer

Ejecuta `open\latch doctor` para ver qué archivo no pudo leer, y reinstala o actualiza Cursor.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1415 — El hook de Cursor no llega al daemon

OL-1415 · ERR_CURSOR_HOOK_UNREACHABLE

Qué ha pasado

Cursor ejecutó el hook de OpenLatch, pero el hook no pudo llegar al daemon. Los eventos de Cursor solo llegan al cliente más tarde, desde el registro de respaldo, y cada acción de Cursor continúa sin decisión hasta que se corrija.

Qué hacer

Ejecuta `openlatch start` para asegurarte de que el daemon está en marcha, y luego `openlatch doctor`. Los errores de hooks propios de Cursor están en su canal de salida Hooks.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1416 — Algunos agentes se quedaron sin daemon

OL-1416 · ERR_HOOK_AGENTS_STRANDED

Qué ha pasado

`openlatch uninstall --agent` desconectó solo algunos de los agentes de este host y, como está documentado, aun así detuvo el daemon y guardó la supervisión `mode = disabled` para toda la máquina (#325: un daemon en ejecución interpreta la eliminación como una manipulación y lo reinstala). Cada agente que quedó fuera sigue conectado a un daemon que ya no se ejecuta, así que cada una de sus llamadas de hook falla en abierto hacia `fallback.jsonl` y no se aplica nada.

Qué hacer

Ejecuta `openlatch start`, y `openlatch system supervision enable` si el daemon estaba supervisado.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1417 — Plugin del agente pendiente de su primera carga

OL-1417 · ERR_OPENCODE_PLUGIN_PENDING

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. El cliente escribió el archivo de plugin del agente y desde entonces no se ha notificado ninguna carga. El agente solo carga plugins cuando arranca su servidor, así que la comprobación queda pendiente hasta entonces.

Qué hacer

Reinicia el agente y ejecuta `open\latch doctor` para confirmar que la comprobación se ha resuelto.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1418 — Otro archivo ocupa la ruta del plugin

OL-1418 · ERR_OPENCODE_PLUGIN_FOREIGN

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. Un archivo que OpenLatch no escribió está donde el cliente escribiría el plugin del agente, y el cliente nunca lo sobrescribe.

Qué hacer

Mueve ese archivo a otro lugar y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1419 — Versión del agente por debajo del mínimo admitido

OL-1419 · ERR_OPENCODE_VERSION_UNSUPPORTED

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. La versión instalada del agente es anterior al mínimo que espera el cliente.

Qué hacer

Actualiza el agente a una versión reciente y ejecuta `open\latch doctor`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1420 — Estado de un componente del agente desconocido

OL-1420 · ERR_OPENCODE_SURFACE_UNDETERMINED

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. El agente está presente, pero el cliente no pudo establecer el estado de uno de sus componentes.

Qué hacer

Ejecuta `openlatch doctor` para ver qué componente no pudo comprobar.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1421 — El plugin del agente no llega al daemon

OL-1421 · ERR_OPENCODE_HOOK_UNREACHABLE

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. Falló la comprobación de que el plugin del agente puede llegar al daemon.

Qué hacer

Ejecuta `openlatch doctor --fix` y reinicia el agente.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1422 — El plugin del agente no se cargó

OL-1422 · ERR_OPENCODE_PLUGIN_NOT_LOADED

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. El agente envió llamadas al modelo por el relé, pero su plugin nunca notificó que se cargara, así que puede que se haya omitido.

Qué hacer

Reinicia el agente y comprueba que no se inicia con `--pure`, que omite todos los plugins.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1423 — Relé del agente aún sin comprobar

OL-1423 · ERR_OPENCODE_RELAY_UNPROVEN

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. El relé de modelo está conectado para el agente, pero todavía no se ha registrado ninguna de sus llamadas al modelo.

Qué hacer

Ejecuta el agente una vez desde un shell que cargue el entorno de OpenLatch (`env . sh`, o `env . ps1` en PowerShell).

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1424 — Superficie del agente no cubierta por el relé

OL-1424 · ERR_OPENCODE_RELAY_UNSUPPORTED

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. Una superficie o un transporte del agente no puede pasar por el relé de modelo, así que sus llamadas al modelo van directamente al proveedor. La salida de doctor indica el motivo.

Qué hacer

Ejecuta `openlatch doctor` para ver qué superficie está afectada.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1425 — La herramienta de línea de comandos de un agente evita el relé

OL-1425 · ERR_CLI_SHIM_SHADOWED

Qué ha pasado

En Windows, la herramienta de línea de comandos de un agente conectado, como la CLI de Cursor o de Cline, se resuelve mediante el PATH de la máquina a un ejecutable distinto del lanzador de OpenLatch. Una instalación para toda la máquina oculta el lanzador, así que las llamadas al modelo de esa herramienta van directamente al proveedor. Doctor solo lee rutas y metadatos de archivos: nunca ejecuta el ejecutable y nunca cambia el PATH de la máquina.

Qué hacer

Elimina el ejecutable que indica `openlatch doctor`, o pide a un administrador que mueva la entrada del PATH de la máquina detrás de la del Path del usuario. Después vuelve a ejecutar `openlatch doctor`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1426 — Aplicación de escritorio del agente aún sin registrar

OL-1426 · ERR_OPENCODE_DESKTOP_SERVICE_UNARMED

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. El servicio en segundo plano compartido del agente se ejecuta sin el relé de modelo, así que las llamadas al modelo de la aplicación de escritorio todavía no se registran. El agente sigue funcionando. La salida de doctor indica la causa.

Qué hacer

Cierra la aplicación de escritorio del agente y todas sus sesiones de terminal, y vuelve a abrir la aplicación de escritorio.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1427 — El servicio de escritorio del agente no pudo iniciarse

OL-1427 · ERR_OPENCODE_DESKTOP_SERVICE_FAILED

Qué ha pasado

Este código procede de la integración del cliente con OpenCode, un agente que aún no es compatible. El daemon encontró la herramienta de línea de comandos incluida en la aplicación de escritorio del agente, pero no pudo iniciar ni registrar el servicio en segundo plano del agente, o el servicio se reinició demasiadas veces. El agente sigue funcionando.

Qué hacer

Ejecuta `openlatch doctor`: indica el ejecutable y el error. Corrige lo que indica y luego ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1500 — Puerto ya en uso

OL-1500 · ERR_PORT_IN_USE

Qué ha pasado

El puerto seleccionado ya lo está usando otro proceso.

Qué hacer

Libera el puerto, o elige otro con `OPENLATCH_PORT`, y ejecuta `openlatch start`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1501 — El daemon ya está en marcha

OL-1501 · ERR_ALREADY_RUNNING

Qué ha pasado

Ya hay una instancia del daemon en ejecución en esta máquina.

Qué hacer

Ejecuta `openlatch status` para ver el daemon en marcha, o `openlatch restart` para reiniciarlo.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1502 — El daemon no llegó a estar operativo

OL-1502 · ERR_DAEMON_START_FAILED

Qué ha pasado

El proceso del daemon se inició, pero la comprobación de salud falló dentro del tiempo límite. También es lo que informa `openlatch update` cuando el daemon que reinició en una versión nueva no volvió, falló su comprobación de salud tras el reinicio o se revirtió a la versión anterior.

Qué hacer

Ejecuta `openlatch doctor` para ver por qué el daemon no está operativo, y luego `openlatch start`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1503 — Nueva versión del cliente disponible

OL-1503 · ERR_VERSION_OUTDATED

Qué ha pasado

Hay una versión más reciente de openlatch disponible (advertencia, no bloqueante).

Qué hacer

Ejecuta `openlatch update` para instalarla.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1504 — No se pudo aplicar la actualización

OL-1504 · ERR_UPDATE_VERIFY_FAILED

Qué ha pasado

La actualización no terminó: falló la comprobación o la descarga, o los archivos descargados no superaron una comprobación de integridad o de arranque. Antes del intercambio no cambia nada en el disco. Si falló el propio intercambio, quedan copias de seguridad junto a los binarios para la restauración.

Qué hacer

Vuelve a ejecutar `openlatch update`. Si falla una segunda vez, ejecuta `openlatch doctor --rescue` y envía el archivo al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1505 — Compilado desde el código fuente, sin actualización automática

OL-1505 · ERR_UPDATE_REFUSED_CARGO_INSTALL

Qué ha pasado

Este cliente se compiló desde el código fuente con cargo `install`, así que la actualización integrada no puede sustituirlo.

Qué hacer

Vuelve a compilarlo desde el código fuente para actualizarlo, o reinstálalo con uno de los comandos de la página de instalación para que la actualización integrada pueda gestionarlo.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1506 — El daemon no respondió a la actualización

OL-1506 · ERR_UPDATE_DAEMON_UNREACHABLE

Qué ha pasado

La CLI esperaba comunicarse con el endpoint `/admin/update` de un daemon en ejecución, pero no pudo alcanzarlo. Distinto de OL-1500 (el daemon del `*usuario*` no pudo enlazar un puerto al arrancar). También es lo que informa `openlatch update` cuando el daemon al que sigue deja de informar su progreso: la misma etapa y el mismo número de bytes durante dos minutos. La actualización puede terminar aún; ejecuta `openlatch status` para ver qué versión está en servicio.

Qué hacer

Ejecuta `openlatch status` para ver qué versión está en marcha.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1507 — No se pudo detener el daemon

OL-1507 · ERR_DAEMON_STOP_FAILED

Qué ha pasado

`openlatch stop/restart` no pudo terminar el daemon: sobrevivió al `/shutdown` ordenado, a `SIGTERM` y a `SIGKILL` (en la práctica nunca ocurre: un proceso que sobrevive a `SIGKILL` está atascado en una espera del kernel no interrumpible). El usuario debe limpiar manualmente. Distinto de OL-1300 (un error de valor de *configuración*), que esta ruta usaba antes por error.

Qué hacer

Termina el proceso con el administrador de tareas de tu sistema, o reinicia la máquina, y ejecuta `openlatch start`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1510 — Una parte del daemon se reinició

OL-1510 · ERR_TASK_PANICKED

Qué ha pasado

Una tarea supervisada dentro del proceso entró en pánico. El supervisor la capturó mediante el `JoinHandle` de la tarea y reinicia el subsistema tras una espera; el resto del daemon sigue funcionando. Se registra con nivel `ERROR` la primera vez y con nivel `DEBUG` en cada repetición consecutiva, para que un subsistema que entra en pánico en cada ventana de espera no pueda inundar `daemon.log`.

Qué hacer

No hace falta hacer nada si se recupera. Si se repite, ejecuta `openlatch doctor --rescue` y envía el archivo al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1511 — Una parte del daemon sigue fallando

OL-1511 · ERR_TASK_RESTART_LIMIT

Qué ha pasado

Una tarea supervisada falló en cada reinicio durante

`[crate::supervision::task::RESTART_LIMIT_WARN_AT]` intentos consecutivos. El supervisor sigue reintentando —nunca se rinde—, pero la racha se informa una vez para que un subsistema roto de forma permanente sea visible sin seguir los registros DEBUG.

Qué hacer

Ejecuta `openlatch doctor --rescue` y envía el archivo de diagnóstico al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1512 — Daemon degradado

OL-1512 · ERR_SUBSYSTEM_DEGRADED

Qué ha pasado

`/health` informa al menos un subsistema `RestartPolicy::Always` que no está running. El daemon responde a las solicitudes, pero no hace todo lo que anuncia: status pasa a `degraded` en lugar de `ok`.

Qué hacer

Ejecuta `openlatch doctor` para ver qué parte está detenida, y luego `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1513 — Nada reinicia el daemon

OL-1513 · ERR_NO_SUPERVISOR

Qué ha pasado

No hay ningún supervisor del sistema instalado y nadie lo rechazó, así que nada reiniciará el daemon si se detiene o si la máquina se reinicia.

Qué hacer

Ejecuta `openlatch system supervision install`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1600 — No se encontró ninguna credencial

OL-1600 · ERR_NO_CREDENTIALS

Qué ha pasado

No se encontraron credenciales en el llavero, la variable de entorno ni el archivo cifrado.

Qué hacer

Ejecuta `openlatch system auth login`, o define `OPENLATCH_API_KEY` y ejecuta `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1601 — Credencial caducada

OL-1601 · ERR_TOKEN_EXPIRED

Qué ha pasado

La clave de API o el token caducó o se revocó.

Qué hacer

Ejecuta `openlatch system auth login` para volver a iniciar sesión.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1602 — Llavero no disponible

OL-1602 · ERR_KEYCHAIN_UNAVAILABLE

Qué ha pasado

El servicio de llavero del sistema no está disponible (por ejemplo, no hay Secret Service en un Linux sin interfaz gráfica).

Qué hacer

No hace falta hacer nada si `openlatch system auth status` indica que has iniciado sesión. Si no, ejecuta `openlatch system auth login`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1603 — El llavero denegó el acceso

OL-1603 · ERR_KEYCHAIN_PERMISSION

Qué ha pasado

El llavero del sistema denegó el acceso (error de permisos).

Qué hacer

Permite el cliente en los ajustes de tu llavero y ejecuta `openlatch system auth login`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1604 — Falló el archivo de credencial cifrado

OL-1604 · ERR_FILE_FALLBACK_ERROR

Qué ha pasado

Falló el respaldo en archivo cifrado (error de descifrado, archivo ausente o datos dañados).

Qué hacer

Ejecuta `openlatch system auth login` para guardar de nuevo la credencial.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1605 — Se agotó el tiempo de inicio de sesión

OL-1605 · ERR_AUTH_TIMEOUT

Qué ha pasado

El flujo de inicio de sesión expiró tras 5 minutos esperando la respuesta del navegador.

Qué hacer

Vuelve a ejecutar `openlatch system auth login` y completa el inicio de sesión en el navegador que se abre.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1606 — Falló el inicio de sesión

OL-1606 · ERR_AUTH_FLOW_FAILED

Qué ha pasado

Falló el flujo de inicio de sesión (error del servidor de retorno, solicitud truncada, error al abrir el navegador).

Qué hacer

Vuelve a ejecutar `openlatch system auth login`. Si no se puede abrir ningún navegador en esta máquina, define `OPENLATCH_API_KEY` y ejecuta `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1607 — No se pudo revocar la clave al cerrar sesión

OL-1607 · ERR_AUTH_REVOCATION_FAILED

Qué ha pasado

Falló la revocación de la clave de API en el servidor durante el cierre de sesión (solo advertencia: la limpieza local continúa).

Qué hacer

Revoca la clave en la consola en `app.openLatch.ai` si ya no debe funcionar.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1608 — Hay que elegir una organización

OL-1608 · ERR_ORG_CHOICE_REQUIRED

Qué ha pasado

Tu cuenta pertenece a dos o más organizaciones, no se nombró ninguna y no había nadie en un terminal para elegir.

Qué hacer

Nombra la organización con `--org` o `OPENLATCH_ORG`, o vuelve a ejecutar el comando en un terminal y elígela.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1609 — El selector de organización quedó sin respuesta

OL-1609 · ERR_ORG_NOT_CHOSEN

Qué ha pasado

El selector de organización no recibió respuesta: la entrada terminó, se agotó el tiempo o no se pudo leer.

Qué hacer

Vuelve a ejecutar el comando y elige una organización, o nómbrala con `--org` o `OPENLATCH_ORG`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1610 — No se encontró la organización

OL-1610 · ERR_ORG_NOT_FOUND

Qué ha pasado

La organización nombrada con `--org` o `OPENLATCH_ORG` no coincide con ninguna de tus organizaciones.

Qué hacer

El error lista tus organizaciones con sus slugs. Vuelve a ejecutar el comando con uno de esos slugs en `--org` o `OPENLATCH_ORG`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1611 — El nombre de la organización es ambiguo

OL-1611 · ERR_ORG_AMBIGUOUS

Qué ha pasado

El nombre dado con `--org` o `OPENLATCH_ORG` coincide con dos o más de tus organizaciones.

Qué hacer

Nombra la organización por su slug, que ninguna otra organización comparte, o ejecuta el comando en un terminal sin nombrar ninguna y elígela de la lista.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1612 — La clave de API pertenece a otra organización

OL-1612 · ERR_ORG_KEY_MISMATCH

Qué ha pasado

OPENLATCH_API_KEY pertenece a una organización distinta de la nombrada.

Qué hacer

Usa una clave de API de la organización nombrada, o nombra la organización a la que pertenece la clave.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1613 — No se pudo confirmar la organización

OL-1613 · ERR_ORG_UNVERIFIED

Qué ha pasado

Se nombró una organización y no se pudo contactar con la plataforma para confirmarla.

Qué hacer

Ejecuta `openlatch system proxy test` para comprobar la ruta de esta máquina a la plataforma y vuelve a ejecutar el comando.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1614 — La máquina está inscrita en otra organización

OL-1614 · ERR_ALREADY_ENROLLED

Qué ha pasado

Esta máquina está inscrita en otra organización. Solo `openlatch system auth switch` la mueve.

Qué hacer

Ejecuta `openlatch system auth switch` para mover esta máquina a la organización que quieras.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1615 — No se aplicó el cambio de organización

OL-1615 · ERR_SWITCH_DAEMON

Qué ha pasado

El daemon no aceptó el cambio de organización. No cambió nada.

Qué hacer

Ejecuta `openlatch doctor` para comprobar el daemon y vuelve a ejecutar `openlatch system auth switch`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1616 — Cambio rechazado: la clave viene del entorno

OL-1616 · ERR_SWITCH_ENV_KEY

Qué ha pasado

`openlatch system auth switch` se rechazó porque la clave viene de `OPENLATCH_API_KEY`.

Qué hacer

Define `OPENLATCH_API_KEY` con una clave de la organización que quieras y ejecuta `openlatch init`, o elimina la variable y ejecuta `openlatch system auth switch`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1617 — El código de inicio de sesión caducó

OL-1617 · ERR_SIGN_IN_EXPIRED

Qué ha pasado

El código de inicio de sesión caducó o ya se usó.

Qué hacer

Vuelve a ejecutar `openlatch system auth login` y completa el inicio de sesión en el navegador que se abre.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1618 — La cuenta no tiene organización

OL-1618 · ERR_NO_ORGANIZATION

Qué ha pasado

Tu cuenta no pertenece a ninguna organización disponible.

Qué hacer

Pide a un administrador de tu organización que te invite y vuelve a iniciar sesión.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1619 — La organización no está disponible

OL-1619 · ERR_ORG_UNAVAILABLE

Qué ha pasado

La organización elegida está suspendida o eliminada, o su licencia terminó.

Qué hacer

Elige otra organización o contacta con el administrador de tu organización.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1620 — Demasiados inicios de sesión en curso

OL-1620 · ERR_SIGN_IN_THROTTLED

Qué ha pasado

Hay demasiados inicios de sesión en curso para esta cuenta.

Qué hacer

Espera unos minutos y vuelve a ejecutar `openlatch system auth login`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1700 — Archivo de consentimiento de telemetría dañado

OL-1700 · ERR_TELEMETRY_CONFIG_CORRUPT

Qué ha pasado

El archivo de consentimiento `telemetry.json` contiene JSON no válido y no se puede analizar.

Qué hacer

Guarda de nuevo tu elección con `openlatch system telemetry enable` o `openlatch system telemetry disable`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1701 — No se puede escribir el consentimiento de telemetría

OL-1701 · ERR_TELEMETRY_WRITE_FAILED

Qué ha pasado

No se pudo escribir el archivo de consentimiento `telemetry.json` (permisos o error de E/S).

Qué hacer

Corrige los permisos de `~/.openlatch` y guarda de nuevo tu elección con `openlatch system telemetry disable` o `openlatch system telemetry enable`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1702 — Lote de telemetría no enviado

OL-1702 · ERR_TELEMETRY_POST_FAILED

Qué ha pasado

Falló un POST de un lote de telemetría (informativo: eventos descartados, sin reintento).

Qué hacer

No hace falta hacer nada. Para dejar de enviar telemetría de uso, ejecuta `openlatch system telemetry disable`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1703 — No se pudo iniciar la telemetría

OL-1703 · ERR_TELEMETRY_INIT_FAILED

Qué ha pasado

El subsistema de telemetría no se pudo inicializar (falta la clave integrada, error de canal).

Qué hacer

No hace falta hacer nada. Ejecuta `openlatch system telemetry status` para ver el estado actual.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1704 — Entrega de telemetría rechazada

OL-1704 · ERR_TELEMETRY_HANDOFF_REJECTED

Qué ha pasado

El daemon rechazó el envío de telemetría de un proceso de la CLI (POST /admin/telemetry/events): el cuerpo superaba el límite de bytes, o no era un lote de eventos de telemetría dentro del límite de eventos. No se retransmitió nada y no se reintenta nada: los eventos se descartan (14). Solo lo ve otro proceso, y la CLI lo ignora; el operador no tiene nada que hacer.

Qué hacer

No hace falta hacer nada.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1800 — Registro de reparación de doctor dañado

OL-1800 · ERR_DOCTOR_JOURNAL_CORRUPT

Qué ha pasado

El archivo de registro de doctor `--fix` existe, pero no se puede analizar (`--restore` no puede usarlo).

Qué hacer

Ejecuta `openlatch doctor` para comprobar el estado actual, y `openlatch init` para reinstalar limpiamente si hace falta.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1801 — Nada que restaurar

OL-1801 · ERR_DOCTOR_RESTORE_NO_JOURNAL

Qué ha pasado

Se invocó `doctor --restore` sin ningún registro previo de `--fix` en disco.

Qué hacer

No hace falta hacer nada. Ejecuta `openlatch doctor` para comprobar el estado actual.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1802 — No se puede escribir el archivo de diagnóstico

OL-1802 · ERR_DOCTOR_RESCUE_WRITE_FAILED

Qué ha pasado

`doctor --rescue` no pudo escribir el archivo comprimido (disco lleno, permisos, etc.).

Qué hacer

Libera espacio en disco o corrige los permisos, y vuelve a ejecutar `openlatch doctor --rescue`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1803 — Archivo de diagnóstico incompleto

OL-1803 · ERR_DOCTOR_RESCUE_PARTIAL

Qué ha pasado

doctor --rescue generó un archivo comprimido, pero uno o varios recolectores dieron error.

Qué hacer

Envía el archivo al soporte tal como está; indica qué partes faltan.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1900 — Clave de integridad de los hooks no disponible

OL-1900 · ERR_HMAC_KEY_UNAVAILABLE

Qué ha pasado

Clave HMAC no disponible: fallaron tanto el llavero del sistema como el respaldo en archivo.

Qué hacer

Ejecuta `openlatch doctor` para comprobar el llavero, y luego `openlatch init` para reinstalar.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1901 — Archivo de estado de los hooks dañado

OL-1901 · ERR_STATE_FILE_CORRUPT

Qué ha pasado

El archivo `hook-state.json` existe, pero no se puede analizar (JSON no válido o `schema_version` desconocida).

Qué hacer

Ejecuta `openlatch init` para reinstalar los hooks y volver a escribir el archivo.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1902 — No se puede escribir el archivo de estado de los hooks

OL-1902 · ERR_STATE_FILE_WRITE_FAILED

Qué ha pasado

Falló la escritura atómica de hook-state.json después de escribir settings.json.

Qué hacer

Corrige los permisos de ~/.openlatch y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1903 — Versión del marcador de hook no admitida

OL-1903 · ERR_MARKER_SCHEMA_UNSUPPORTED

Qué ha pasado

El campo `_openlatch.v` lleva una versión que no entendemos.

Qué hacer

Actualiza el cliente con `openlatch update` y ejecuta `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1904 — No se pudo procesar una entrada de hook

OL-1904 · ERR_CANONICALIZATION_FAILED

Qué ha pasado

La canonicalización JCS rechazó la entrada de hook (error interno; no debería ocurrir).

Qué hacer

Ejecuta `openlatch doctor --rescue` y envía el archivo de diagnóstico al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1905 — Marcador de hook antiguo actualizado

OL-1905 · ERR_LEGACY_MARKER_DETECTED

Qué ha pasado

Se encontró un marcador booleano heredado `_open\latch: true` y se actualizará (nivel informativo).

Qué hacer

No hace falta hacer nada.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1906 — Hooks modificados reparados

OL-1906 · ERR_TAMPER_DETECTED

Qué ha pasado

Las entradas de hook se modificaron fuera de OpenLatch y el reconciliador las reparó.

Qué hacer

Busca la herramienta o el script que modifica el archivo de ajustes del agente, y ejecuta `openlatch system hooks status` para confirmar que los hooks se mantienen.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1910 — No se puede sustituir el archivo de ajustes

OL-1910 · ERR_ATOMIC_WRITE_FAILED

Qué ha pasado

El cliente no pudo escribir la nueva versión de un archivo de ajustes del agente en lugar de la anterior, por un error de permisos o de disco. El archivo original no ha cambiado.

Qué hacer

Corrige los permisos de la carpeta de ajustes del agente y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1911 — No se puede resolver el enlace del archivo de ajustes

OL-1911 · ERR_SYMLINK_CANONICALIZE_FAILED

Qué ha pasado

Un archivo de ajustes del agente es un enlace simbólico, y el cliente no pudo resolver adónde apunta.

Qué hacer

Comprueba que el destino del enlace existe y se puede leer, y vuelve a ejecutar `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1950 — Supervisión no admitida en este sistema

OL-1950 · ERR_SUPERVISION_UNSUPPORTED_OS

Qué ha pasado

El cliente no puede instalar un supervisor del sistema operativo en este sistema, así que nada reinicia el daemon automáticamente.

Qué hacer

Inicia tú mismo el daemon con `openlatch start` después de cada reinicio.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1951 — No se pudo instalar el supervisor

OL-1951 · ERR_SUPERVISION_INSTALL_FAILED

Qué ha pasado

El cliente no pudo instalar el supervisor del sistema operativo (launchd, systemd o Task Scheduler): falló la escritura de sus archivos, o el supervisor rechazó el registro.

Qué hacer

Lee el error del sistema operativo que se muestra con el código, que nombra la causa: un permiso de carpeta, un fallo de `systemctl` o la falta de un bus de sesión de usuario en Linux, o un fallo de `schtasks` en Windows. Corrígelo y después ejecuta `openlatch system supervision install`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1952 — No se pudo registrar el supervisor

OL-1952 · ERR_SUPERVISION_BOOTSTRAP_FAILED

Qué ha pasado

En macOS, launchd se negó a registrar el supervisor del cliente, por ejemplo porque no hay ninguna sesión de usuario conectada por SSH.

Qué hacer

Ejecuta `openlatch system supervision install` desde una sesión de escritorio en la máquina.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-1953 — El supervisor rechazó un comando

OL-1953 · ERR_SUPERVISION_CONTROL_FAILED

Qué ha pasado

Falló un `start` / `stop` / `restart` entregado al supervisor del sistema: la unidad está registrada, pero `systemctl` / `launchctl` / `schtasks` rechazó el verbo de control. Los llamadores recurren a actuar directamente sobre el proceso, el comportamiento anterior a la supervisión, así que es una ruta degradada y no un fallo grave.

Qué hacer

Ejecuta `openlatch system supervision status` para comprobar el supervisor.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2000 — No se pudo iniciar el seguimiento de configuración

OL-2000 · ERR_INVENTORY_INIT_FAILED

Qué ha pasado

Falló la inicialización del monitor de configuración (error al analizar el manifiesto o al preparar el observador).

Qué hacer

Ejecuta `openlatch system inventory status` para ver la causa, y luego `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2001 — No se encontró la lista de fuentes de configuración

OL-2001 · ERR_INVENTORY_MANIFEST_NOT_FOUND

Qué ha pasado

No se encontró el archivo de manifiesto en la ruta configurada.

Qué hacer

Ejecuta `openlatch init` para restaurarlo.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2002 — Lista de fuentes de configuración no válida

OL-2002 · ERR_INVENTORY_MANIFEST_PARSE

Qué ha pasado

Falló la validación del esquema del manifiesto.

Qué hacer

Ejecuta `openlatch update` y luego `openlatch init` para restaurar una lista válida.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2003 — No se puede expandir una ruta seguida

OL-2003 · ERR_INVENTORY_PATH_EXPANSION

Qué ha pasado

Falló la expansión de ruta (variable desconocida, variable de entorno ausente, fallo de una llamada al sistema).

Qué hacer

Busca OL-2003 en `daemon.log`, en la carpeta `logs` del directorio de estado del cliente: cada entrada nombra la ruta que no se pudo expandir. Corrige la variable que usa y después ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2004 — No se puede seguir una ruta

OL-2004 · ERR_INVENTORY_WATCHER_FAILED

Qué ha pasado

Falló la inicialización del observador para una ruta concreta (sigue funcionando para las demás rutas).

Qué hacer

Busca OL-2004 en `daemon.log`, en la carpeta `logs` del directorio de estado del cliente: cada entrada nombra la ruta que no se puede vigilar. Comprueba los permisos de esa ruta.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2005 — No se pudo calcular la huella de un archivo de configuración

OL-2005 · ERR_INVENTORY_HASH_FAILED

Qué ha pasado

Falló la canonicalización JCS o la normalización NFC (fail-open: registra y continúa).

Qué hacer

No hace falta hacer nada. Si se repite con el mismo archivo, ejecuta `openlatch system inventory inspect` sobre esa fuente.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2006 — Nuevo análisis de configuración parcial

OL-2006 · ERR_INVENTORY_RESCAN_PARTIAL

Qué ha pasado

Reexploración periódica parcial: algunas rutas no son accesibles; la reexploración termina para el resto.

Qué hacer

Busca OL-2006 en `daemon.log`, en la carpeta `logs` del directorio de estado del cliente: cada entrada nombra una ruta que el nuevo análisis no pudo leer. Comprueba los permisos de esas rutas.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2007 — Detalles de inicio de sesión no disponibles

OL-2007 · ERR_INVENTORY_ENRICH_FAILED

Qué ha pasado

Falló el enriquecimiento al inicio de la sesión (registra una advertencia y continúa con el evento sin enriquecer).

Qué hacer

No hace falta hacer nada. Si se repite, ejecuta `openlatch system inventory status`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-2008 — Seguimiento de configuración desactivado

OL-2008 · ERR_INVENTORY_DISABLED

Qué ha pasado

El monitor del plano de configuración está desactivado en la configuración, así que la deriva de configuración en este host no se observa ni se informa.

Qué hacer

Quita `OPENLATCH_INVENTORY_ENABLED` (o defínela como `true`), y define `enabled = true` bajo `[inventory_monitor]` en `config.toml` (o quita esa línea), y después ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-9999 — Error interno inesperado

OL-9999 · ERR_BUG

Qué ha pasado

Código asignado a todos los errores internos o inesperados que pasan por `bug_report()`.

Qué hacer

Ejecuta `openlatch doctor --rescue` y envía el archivo de diagnóstico al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-CA — Certificado del relé de modelo no fiable

OL-RELAY-CA · ERR_MODEL_RELAY_CA_UNTRUSTED

Qué ha pasado

La CA del relay no es de confianza para el almacén de confianza del sistema del usuario —la instalación se omitió (sin sesión de escritorio), se rechazó (diálogo cancelado) o falló la prueba de cadena o de almacén—, así que no se escribió ningún valor de proxy y el agente va directo (sin captura). Solución: `openlatch doctor --fix` desde una sesión de escritorio, o el comando de instalación de la plataforma. Se muestra bajo `PreflightFailed`.

Qué hacer

Ejecuta `openlatch doctor --fix` desde una sesión de escritorio en la máquina.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-CURSOR-UNPROVEN — Relé de Cursor aún sin comprobar

OL-RELAY-CURSOR-UNPROVEN · ERR_CURSOR_RELAY_UNPROVEN

Qué ha pasado

El relay de Cursor está conectado y desde entonces no le ha llegado ninguna solicitud de Cursor descifrada (`agents.cursor.relay.traffic = "unproven"` en `doctor`). Una sola prueba para todo el agente: el relay no puede distinguir las solicitudes de la CLI de las del IDE. Solución: inicia un turno de agente de Cursor.

Qué hacer

Inicia un turno del agente de Cursor y ejecuta `openlatch doctor`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-CURSOR-UNSUPPORTED — Superficie de Cursor no cubierta por el relé

OL-RELAY-CURSOR-UNSUPPORTED · ERR_CURSOR_RELAY_UNSUPPORTED

Qué ha pasado

Una superficie de Cursor que esta versión no puede dirigir al relay (`agents.cursor.relay.{cli,ide} = "unsupported"`): el IDE, hasta que se supere su prueba. Sus llamadas al modelo van directas y no se capturan; el plano de comandos sigue aplicando las reglas. Tampoco figura en `[ERR_MODEL_RELAY_CODES]`.

Qué hacer

Mantén el cliente al día con `openLatch update`; la superficie queda cubierta cuando una versión la añade.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-DELIVERY — La ruta del agente hacia el relé no está lista

OL-RELAY-DELIVERY · ERR_MODEL_RELAY_NOT_DELIVERED

Qué ha pasado

El relé de modelo está en marcha y comprobado, pero la ruta propia del agente hacia él no está lista: una clave de configuración apunta a otro proxy o no se puede leer, falta un entorno de lanzador, o el runtime de Node del agente no tiene ningún certificado en el que confiar. Las llamadas al modelo del agente no se registran hasta que se resuelva. La salida de doctor indica cuál.

Qué hacer

Ejecuta `openlatch doctor --fix` y luego reinicia el agente. Si `openlatch doctor` sigue indicando este código, corrige lo que indica.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-DOWN — El relé de modelo no está en marcha

OL-RELAY-DOWN · ERR_MODEL_RELAY_NOT_RUNNING

Qué ha pasado

Nada responde en el endpoint de estado del model relay en el puerto fijado (conexión rechazada).

Qué hacer

Ejecuta `openlatch restart` y luego `openlatch system model-relay status`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-FOREIGN — Otro proceso ocupa el puerto del relé

OL-RELAY-FOREIGN · ERR_MODEL_RELAY_PORT_FOREIGN

Qué ha pasado

El puerto fijado del model relay lo ocupa un proceso que NO es el model relay de OpenLatch (firma de estado incorrecta o ausente). Lo informa `openlatch doctor`: un agente conectado a un listener loopback ajeno está enviando la credencial de su proveedor a ese proceso.

Qué hacer

Detén el proceso que ocupa el puerto y ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-IO — Error de entrada o salida del relé de modelo

OL-RELAY-IO · ERR_MODEL_RELAY_IO

Qué ha pasado

Error genérico de E/S del model relay (lo usa el puente `From<std::io::Error>`).

Qué hacer

Ejecuta `openlatch system model-relay status`. Si el error se repite, ejecuta `openlatch doctor --rescue` y envía el archivo al soporte.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-MISCONFIG — Ajuste del relé sustituido

OL-RELAY-MISCONFIG · ERR_MODEL_RELAY_MISCONFIGURED

Qué ha pasado

Está demostrado que el editor cargó la URL del relay de una posición de proveedor y usa ese proveedor, pero no llega tráfico al endpoint: algo por encima del archivo (la configuración remota de una organización) la sobrescribe.

Qué hacer

Revisa la dirección de ese proveedor en la configuración remota o gestionada del editor, y ejecuta `openlatch doctor`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-NEXTSTART — El relé se aplica en el próximo inicio del editor

OL-RELAY-NEXTSTART · ERR_MODEL_RELAY_NEXT_START

Qué ha pasado

Una posición de proveedor está conectada a su endpoint del relay, y aún no se ha demostrado que el editor que la lee la use.

Qué hacer

No hace falta hacer nada. La comprobación se resuelve con la primera llamada al modelo que pase por el relé tras reiniciar el editor.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-NOFIND — Ningún registro local con este identificador

OL-RELAY-NOFIND · ERR_MODEL_RELAY_FINDING_NOT_FOUND

Qué ha pasado

`model-relay explain <finding_id>` no encontró ningún bloque de churn coincidente en el almacén de retención local acotado (nunca se generó aquí, o caducó o se depuró).

Qué hacer

Comprueba el identificador y ejecuta el comando en la máquina que generó el registro.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-PORT — Puerto del relé de modelo ocupado

OL-RELAY-PORT · ERR_MODEL_RELAY_PORT_IN_USE

Qué ha pasado

El puerto loopback fijado del model relay ya está en uso. Se informa de forma explícita, nunca con un nuevo sondeo silencioso (D-25): la configuración del agente apunta a ese puerto exacto.

Qué hacer

Libera el puerto e inicia de nuevo el daemon con `openlatch start`. Para funcionar sin el relé de modelo, desactívalo con `openlatch system model-relay disable`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-PORTS — Ningún puerto de relé para un proveedor

OL-RELAY-PORTS · ERR_MODEL_RELAY_ENDPOINT_PORTS

Qué ha pasado

Una posición de proveedor no pudo obtener un puerto de endpoint: su puerto en el bloque de endpoints (`[port + 1, port + 32]` por encima del puerto principal del relay) lo ocupa otro proceso, o todos los puertos del bloque ya están asignados.

Qué hacer

Libera un puerto justo por encima del puerto del relé de modelo y ejecuta `openlatch restart`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-PREFLIGHT — El relé de modelo no llega al proveedor

OL-RELAY-PREFLIGHT · ERR_MODEL_RELAY_PREFLIGHT_FAILED

Qué ha pasado

El model relay enlazó su puerto, pero una solicitud sintética no pudo completar el viaje de ida y vuelta al proveedor a través de él, así que el agente se dejó deliberadamente sin conectar (`model_relay::preflight`). Lo informa `init` —que termina con un código distinto de cero dejando los hooks, el daemon y la supervisión completamente instalados— y `doctor`. NO es una condición fatal para el daemon: sigue funcionando y se vuelve a conectar en cuanto la prueba se supera.

Qué hacer

Ejecuta `openlatch system proxy test` para comprobar la ruta hacia el proveedor, y luego `openlatch doctor`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-PROVIDER — La configuración del agente contiene otra entrada de proveedor

OL-RELAY-PROVIDER · ERR_MODEL_RELAY_FOREIGN_PROVIDER

Qué ha pasado

Ya existe en la configuración del agente una tabla de proveedor con el nombre que escribe OpenLatch y **no** apunta a nuestro listener loopback: es de otra persona. La instalación se niega a sobrescribirla, porque entonces la desinstalación no podría devolver el archivo a su estado original: la reversión eliminaría una tabla que nunca fue nuestra y restauraría un puntero que nunca registramos.

Qué hacer

Cambia el nombre de la tabla [`model_providers.<name>`] en conflicto o elimínala, y ejecuta `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-SERVE — El relé de modelo se detuvo por un error

OL-RELAY-SERVE · ERR_MODEL_RELAY_SERVE

Qué ha pasado

El servidor axum del model relay terminó con un error de E/S mientras atendía solicitudes.

Qué hacer

Ejecuta `openlatch restart` y luego `openlatch system model-relay status`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-SETTINGS — Los ajustes del agente contienen tu propio proxy

OL-RELAY-SETTINGS · ERR_MODEL_RELAY_SETTINGS

Qué ha pasado

El archivo de configuración de un agente contiene, en la clave que usaría OpenLatch, un proxy que OpenLatch no escribió, o no se puede analizar como JSONC. Se deja intacto y esa superficie no se captura. Solución: mueve ese proxy a la configuración [proxy] de OpenLatch, por la que ya pasa el tráfico saliente del relay.

Qué hacer

Lleva ese proxy a la configuración del cliente con `openlatch system proxy set` y ejecuta `openlatch init`.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-STATEFILE — No se puede editar de forma segura el archivo de estado del agente

OL-RELAY-STATEFILE · ERR_MODEL_RELAY_STATE_FILE

Qué ha pasado

Un archivo de estado de agente que OpenLatch edita para conectar una posición de proveedor no se pudo editar con seguridad: supera el límite de tamaño con el que se lee, o no es JSON válido.

Qué hacer

Ejecuta `openlatch doctor` para saber de qué archivo se trata, y corrígelo o redúcelo.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-TRUSTPROOF — Confianza del relé aún sin comprobar aquí

OL-RELAY-TRUSTPROOF · ERR_MODEL_RELAY_TRUST_UNPROVEN

Qué ha pasado

El almacén de confianza del usuario se verificó en una plataforma cuya ruta de confianza aún no tiene prueba en una máquina real (Windows, Linux). `State :: Degraded` con `Resolution :: NothingToDo`: un límite de cobertura (D-30).

Qué hacer

No hace falta hacer nada.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

OL-RELAY-UNCOVERED — Proveedor no cubierto por el relé

OL-RELAY-UNCOVERED · ERR_MODEL_RELAY_UNCOVERED

Qué ha pasado

Un proveedor configurado no puede pasar por el relay, por un motivo concreto: sus solicitudes están firmadas sobre el host, esta versión no sabe adónde las enviaría de otro modo, o su cliente no expone ningún ajuste que redirigir.

Qué hacer

No hace falta hacer nada. Ejecuta `openlatch doctor` para ver qué proveedores no están cubiertos.

Nota

Esta página se genera a partir del código fuente del cliente en un commit fijado. El mensaje de arriba es el que muestra el cliente.

Cliente 0.6.19

13 de octubre de 2026 · Cliente

`openlatch init` y `openlatch doctor` indican si las llamadas al modelo de cada agente están listas para registrarse, y la instalación en Windows pide menos.

- `openlatch init` y `openlatch doctor` indican, para cada agente, si sus llamadas al modelo están listas para registrarse, y qué falta todavía.
- `openlatch init` pregunta antes de reiniciar un agente que ya estaba en marcha, y nombra los que te deja reiniciar a ti.
- Un Cursor que ya estaba abierto durante `openlatch init` queda cubierto en cuanto lo cierras y lo vuelves a abrir; recargar sus ventanas no basta.
- En Windows, la instalación ya no muestra un aviso del Firewall, y el diálogo de confianza del certificado va precedido de su huella digital para que puedas comprobarla.
- En Windows, correcciones de estabilidad para Cline, para una reinstalación después de `openlatch uninstall --purge`, y para los reinicios.
- Trabajo preparatorio para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.

[@openlatch/client@0.6.19](#)

Cliente 0.6.18

13 de octubre de 2026 · Cliente

Elige tu organización al iniciar sesión y mueve una máquina a otra organización.

- Al iniciar sesión se pregunta qué organización usar cuando tu cuenta pertenece a varias; `--org` o `OPENLATCH_ORG` la nombra sin preguntar.
- El nuevo comando `openlatch system auth switch` mueve esta máquina a otra de tus organizaciones.
- En Windows, `openlatch doctor` informa de la herramienta de línea de comandos de un agente cuya instalación para toda la máquina evita el relé (OL-1425).
- Trabajo preparatorio para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.

[@openlatch/client@0.6.18](#)

Cliente 0.6.17

12 de octubre de 2026 · Cliente

Cursor vuelve a estar cubierto en cuanto se recargan sus ventanas.

- Un Cursor en espera vuelve a estar cubierto en cuanto se recargan sus ventanas.
- El relé registra cada llamada de Cursor una sola vez y clasifica el tráfico de un agente de código que OpenLatch aún no admite.

[@openlatch/client@0.6.17](#)

Cliente 0.6.16

12 de octubre de 2026 · Cliente

El cliente también conecta la aplicación de escritorio de Codex, sin herramienta de línea de comandos y sin un nuevo `openlatch init`.

- El cliente conecta la aplicación de escritorio de Codex en una máquina sin herramienta de línea de comandos de Codex, sin volver a ejecutar `openlatch init`.
- `openlatch doctor` ya no informa de un falso problema para un agente de código que OpenLatch aún no admite.

[@openlatch/client@0.6.16](#)

Una política en varias Autonomy Zones

12 de octubre de 2026 · Plataforma

Una Agent Intent Policy puede aplicarse en varias zonas a la vez, cada vínculo con su propio modo. Antes de vincular, desvincular o cambiar un modo, la consola muestra una vista previa del impacto. Se rechaza un vínculo que relajaría lo que una zona superior ya bloquea.

Cliente 0.6.15

11 de octubre de 2026 · Cliente

En Windows, las herramientas de línea de comandos de Cursor y Cline pasan por el relé sin configuración.

- En Windows, las herramientas de línea de comandos de Cursor y Cline envían su tráfico de modelo a través del relé sin configuración.
- En Windows, la comprobación del certificado del relé ya no lee un archivo que sigue abierto.

[@openlatch/client@0.6.15](#)

Cliente 0.6.14

11 de octubre de 2026 · Cliente

Trabajo preparatorio para un agente de código que OpenLatch aún no admite.

- Trabajo preparatorio del relé para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.

[@openlatch/client@0.6.14](#)

Cliente 0.6.13

10 de octubre de 2026 · Cliente

Trabajo preparatorio para un agente de código que OpenLatch aún no admite, y una corrección del relé en Windows.

- Trabajo preparatorio de hooks e inventario para un agente de código que OpenLatch aún no admite. Nada cambia para los agentes ya conectados.
- En Windows, el certificado del relé se comprueba en el contexto del propio usuario, y los procesos en segundo plano se inician sin abrir una ventana de consola.

[@openlatch/client@0.6.13](#)

Cliente 0.6.12

10 de octubre de 2026 · Cliente

Un uso de modelos más exacto y una mejor lectura de PowerShell.

- `openlatch doctor --fix` informa del estado en que deja el equipo.
- Se conserva el uso que un flujo de modelo interrumpido ya había informado.
- Un modelo se marca como desconocido solo cuando su uso es estimado.
- Una clave de API se distingue de un token de suscripción por su forma.
- Un parámetro de PowerShell escrito como `-Param:value` se lee igual que `-Param value`.

[@openlatch/client@0.6.12](#)

Cliente 0.6.11

9 de octubre de 2026 · Cliente

Los comandos de PowerShell se leen con más precisión.

- Una URL en un comando de PowerShell se lee ahora como una URL, no como una ruta de archivo.

[@openlatch/client@0.6.11](#)

Cliente 0.6.10

8 de octubre de 2026 · Cliente

Correcciones para el registro y para la lectura de comandos de PowerShell.

- La comprobación de estado del daemon en reposo envía ahora la clave de API.
- `openlatch init` guarda ahora un `OPENLATCH_API_URL` exportado en la configuración del cliente.
- Una ruta de archivo se clasifica donde se resuelve, no como está escrita.
- Los comandos de PowerShell se dividen en palabras según las propias reglas de PowerShell.

[@openlatch/client@0.6.10](#)

Autonomy Zones en el árbol de tu organización

8 de octubre de 2026 · Plataforma

Cualquier unidad de tu organización puede tener una Autonomy Zone; una unidad sin ella hereda la zona más cercana por encima. Cada tarjeta de agente muestra su zona y las capas de las que hereda. Puedes asignar agentes a una zona a mano tras previsualizar el cambio, y definir una zona predeterminada para los agentes cuyo propietario aún no está resuelto.

Autonomy Zones y Agent Intent Policies

8 de octubre de 2026 · Plataforma

Cada agente actúa dentro de una Autonomy Zone construida a partir de Agent Intent Policies en cinco dimensiones: Calidad, Economía, Seguridad, Protección y Cumplimiento. Apruebas cada política como una frase generada, con sus casos de prueba generados mostrados al lado. Los veredictos son Allow, Ask, Block u Optimize.

API REST pública

8 de octubre de 2026 · Plataforma

Las operaciones de la consola están disponibles mediante una API REST pública versionada bajo `/api/public/v1`, autenticada con claves de API de organización. Admite claves de idempotencia y límites de frecuencia, y cada error proviene de un catálogo cerrado.

Cliente 0.6.9

7 de octubre de 2026 · Cliente

Informes de configuración en directo, y correcciones para Cline, Cursor y macOS.

- Los cambios de configuración de un equipo se informan en el momento en que ocurren.
- Una sesión compañera de Claude Code se vincula a la sesión que la dirige.
- Las skills, reglas y servidores MCP que cargan Codex CLI, Cursor y Cline se inventarían tal como cada agente los carga realmente.
- Los informes de Cursor nombran el servidor MCP en cada llamada.
- Actualizar Cline.app ya no interrumpe una conversación.
- Una sesión de Claude Code iniciada desde un plan registra ese plan como su primer prompt.
- En macOS, el cliente ya no solicita acceso a las carpetas Escritorio, Descargas y Documentos.
- `openlatch init` ya no describe la telemetría de uso como anónima.
- Las rutas nativas de Windows se clasifican correctamente en las lecturas y escrituras de archivos.

[@openlatch/client@0.6.9](#)

Aislamiento de organizaciones aplicado por la base de datos

6 de octubre de 2026 · Plataforma

La seguridad a nivel de fila está activada y forzada en cada tabla de organización, sin propietario en tiempo de ejecución ni rol de omisión, y se comprueba en CI y en cada arranque. Es un control de ingeniería, no una certificación.

Cliente 0.6.8

6 de octubre de 2026 · Cliente

Correcciones del paquete de políticas y del relé.

- Un paquete de políticas en caché se revalida correctamente cuando la plataforma lo informa sin cambios.
- Ya no se registra el uso de sesiones que ningún hook registró.

[@openlatch/client@0.6.8](#)

Cliente 0.6.7

5 de octubre de 2026 · Cliente

Una evaluación que tiene en cuenta las rutas, y correcciones para Cline.

- Los efectos de una acción se relacionan con las rutas a las que apunta.
- El plugin de Cline se aplica a todas las herramientas, no solo al shell.
- `openlatch init` aguarda la primera entrega del plugin de Cline en lugar de fallar.
- Cada acción se decide una sola vez, no otra vez después de ejecutarse.
- Los hilos secundarios de Codex CLI se archivan bajo su sesión principal.
- El uso se asigna a la instalación correcta.

[@openlatch/client@0.6.7](#)

Políticas que leen lo que hace una acción

5 de octubre de 2026 · Plataforma

Cada acción se clasifica por su efecto y su destino, sea cual sea la herramienta del agente conectado que la lleve: una lectura de secretos, una llamada a un host de red, una escritura en una clase de rutas. Una política escrita una vez se aplica al mismo efecto en todos los agentes.

Cliente 0.6.6

2 de octubre de 2026 · Cliente

Las llamadas de modelo del IDE de Cursor pasan por el relé.

- Las llamadas de modelo del IDE de Cursor se enrutan a través del relé.
- Un `openlatch init` nuevo ya no falla con componentes que aún se están iniciando.
- `openlatch doctor` muestra una línea de estado por agente y reconoce el hub de Cline.app.
- Una sustitución de procesos del shell ya no se lee como destino de una redirección.

[@openlatch/client@0.6.6](#)

Cliente 0.6.5

1 de octubre de 2026 · Cliente

Un vocabulario de herramientas común, y veredictos Ask entregados según cada agente.

- Las herramientas de cada agente se asignan a un vocabulario común; una herramienta sin asignación se registra igualmente con su propio nombre.
- La evaluación de políticas lee un único catálogo de clasificación del paquete de políticas.
- Los veredictos Ask llegan a cada agente en la forma que admite, con un veredicto definido de antemano cuando un agente no tiene un ask propio.

[@openlatch/client@0.6.5](#)

Cliente 0.6.4

1 de octubre de 2026 · Cliente

Compatibilidad con Cursor.

- Las políticas deciden sobre las acciones de Cursor, y el veredicto llega a Cursor en su propio formato.
- El tráfico de modelo de Cursor pasa por el relé y se registra como uso, nunca como contenido.
- La conexión de Cursor se comprueba, se repara y se inventaría como la de cualquier otro agente.
- `openlatch uninstall` nombra los agentes que una desinstalación parcial deja conectados.

[@openlatch/client@0.6.4](#)

Cliente 0.6.3

30 de septiembre de 2026 · Cliente

El plugin de Cline llega a Cline.app y a VS Code.

- El plugin de Cline se entrega a Cline.app y a VS Code.
- El cliente comprueba que el plugin se cargó en cada host.

[@openlatch/client@0.6.3](#)

Licencia y consumo en la consola

29 de septiembre de 2026 · Plataforma

La página de licencia de Configuración compara aquello para lo que tu organización tiene licencia con lo que consume, junto con las fechas del contrato, el historial diario y la lista de hosts, que puedes exportar como informe de despliegue.

Propiedad de los agentes

29 de septiembre de 2026 · Plataforma

Cada agente se resuelve a un propietario: una persona con nombre, un responsable de servicio declarado, o Sin atribuir con su motivo. El cambio de propietario queda registrado en el registro de auditoría, y se señala un agente que sigue funcionando bajo un empleado que ya se fue.

Agentes reconocibles

29 de septiembre de 2026 · Plataforma

Cada tarjeta de agente muestra un nombre que una persona reconoce y cuántas sesiones de cuántos proyectos agrupa. Filtra agentes, inventario y eventos por repositorio, y detecta cuándo una misma máquina se ha convertido en varias filas de agentes.

Grafo de ejecución

29 de septiembre de 2026 · Plataforma

La pestaña Grafo de ejecución dibuja las llamadas de modelo, herramientas, comandos, archivos y llamadas de red de un agente en un periodo. Fíltralo por los caminos más costosos o más arriesgados, abre cualquier nodo para ver los eventos que lo componen, y crea una regla en modo Monitor desde un nodo de comando, reproducida primero sobre el periodo.

Economía de los agentes

29 de septiembre de 2026 · Plataforma

La pestaña Economía de cada agente muestra el gasto, una previsión y el desglose por sesión en el periodo que elijas. Las oportunidades de ahorro se ordenan por su impacto estimado; son un diagnóstico y no cambian nada por sí solas.

Cliente 0.6.2

29 de septiembre de 2026 · Cliente

Informes de Cline, autorreparación y una purga completa.

- Las sesiones, prompts y llamadas a herramientas de Cline se envían a la plataforma.
- El cableado del relé, la confianza de los hooks y los archivos del plugin de Cline se comprueban contra manipulaciones y se reparan.
- El cliente informa cuando un paquete de políticas está pendiente de entrega.
- `openlatch uninstall` envía un evento de desinstalación, y `openlatch uninstall --purge` elimina todo el estado del cliente.

[@openlatch/client@0.6.2](#)

Cliente 0.6.1

29 de septiembre de 2026 · Cliente

Cline a través del relé, y gestión del certificado del relé.

- El tráfico de modelo de Cline se captura a través del relé, incluida la aplicación de escritorio de Cline.
- El relé demuestra que su certificado es de confianza antes de que un agente se dirija a él.
- El certificado del relé se puede consultar, rotar, liberar y eliminar.
- Solo los agentes detectados aparecen en los datos del equipo.
- Los hooks de Codex CLI son de confianza en Windows.

[@openlatch/client@0.6.1](#)

Políticas que siguen decidiendo sin conexión

29 de septiembre de 2026 · Plataforma

Los cambios de políticas llegan a cada cliente en su siguiente consulta, sin una nueva versión del cliente ni acciones en cada equipo. Cada cliente conserva el último paquete recibido y sigue decidiendo con él mientras la red no está disponible. En modo Monitor, lo que una política habría bloqueado se registra con el evento.

Cliente 0.6.0

24 de septiembre de 2026 · Cliente

Un `openlatch update` paso a paso, con un cambio incompatible en su salida JSON.

- Cambio incompatible: `openlatch update --check --json` ya no devuelve `min_supported_client`. `current`, `latest` y `severity` no cambian.
- `openlatch update` muestra su progreso paso a paso, y el daemon vuelve a iniciarse después.
- El inventario de configuración informa de los hooks, reglas y flujos de trabajo de Cline.

[`@openlatch/client@0.6.0`](#)

Cliente 0.5.8

23 de septiembre de 2026 · Cliente

Captura de prompts de Cline.

- Los prompts se capturan en la CLI de Cline y en la aplicación de escritorio de Cline.
- Ya no se informan los tipos de agente que no están conectados.

[@openlatch/client@0.5.8](#)

Cliente 0.5.7

23 de septiembre de 2026 · Cliente

Corrección de la salida del terminal en Windows.

- Los colores y el progreso se muestran correctamente en Windows en todos los comandos, no solo en `openlatch init`.

[@openlatch/client@0.5.7](#)

Cliente 0.5.6

23 de septiembre de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.5.6](#)

Cliente 0.5.4

22 de septiembre de 2026 · Cliente

Correcciones de la supervisión del daemon y del envío de telemetría.

- En Windows, la tarea programada que mantiene el daemon en marcha conserva su marcador de versión.
- El primer lote de telemetría del daemon se envía por HTTPS.

[@openlatch/client@0.5.4](#)

Cliente 0.5.3

22 de septiembre de 2026 · Cliente

La telemetría pasa por la plataforma, y los binarios de Linux arrancan en hosts más antiguos.

- La telemetría y los informes de fallos pasan por el punto de ingesta de la propia plataforma.
- Los binarios de Linux arrancan en hosts mínimos y más antiguos.

[@openlatch/client@0.5.3](#)

Cliente 0.5.2

21 de septiembre de 2026 · Cliente

Correcciones del arranque y del envío de eventos.

- En Windows, openLatch se ejecuta como un programa de consola, y sus binarios arrancan en un equipo recién instalado.
- Se envían los eventos que seguían en cola al apagar.
- El daemon detecta un nuevo inicio de sesión sin reiniciarse.
- El envío de eventos se recupera tras cada reconexión con la plataforma.

[@openlatch/client@0.5.2](#)

Cliente 0.5.1

20 de septiembre de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.5.1](#)

Cliente 0.4.0

8 de septiembre de 2026 · Cliente

Los comandos de los subsistemas pasan a `openlatch system`.

- Cambio incompatible: `openlatch auth`, `hooks`, `model-relay`, `proxy`, `supervision`, `telemetry` e `inventory` pasan a ser `openlatch system <name>`, sin alias.
- Cambio incompatible: la sección de configuración `[boundary]` pasa a ser `[model_relay]`, las variables `OPENLATCH_BOUNDARY_*` pasan a ser `OPENLATCH_MODEL_RELAY_*`, y los códigos de error `OL-BND-*` pasan a ser `OL-RELAY-*`.
- `openlatch doctor` informa del estado en directo de cada hook.
- Los diagnósticos de proxy informan del proxy que el cliente usa realmente.

[@openlatch/client@0.4.0](#)

Cliente 0.3.3

6 de septiembre de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.3.3](#)

Cliente 0.2.2

28 de agosto de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.2.2](#)

Cliente 0.1.18

19 de agosto de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.1.18](#)

Cliente 0.1.14

23 de abril de 2026 · Cliente

Corrección de la dirección de API predeterminada.

- La dirección de API predeterminada ya no repite /api.

[@openlatch/client@0.1.14](#)

Cliente 0.1.13

23 de abril de 2026 · Cliente

El paquete npm se resuelve con `npx @openlatch/client init`.

- El paquete npm incorpora un punto de entrada `client`, de modo que `npx @openlatch/client init` se resuelve.
- La ayuda ya no muestra un banner.

[@openlatch/client@0.1.13](#)

Cliente 0.1.12

23 de abril de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.1.12](#)

Cliente 0.1.8

23 de abril de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.1.8](#)

Cliente 0.1.7

23 de abril de 2026 · Cliente

El paquete npm incluye el punto de entrada del hook.

- El paquete npm instala el punto de entrada openlatch-hook.

[@openlatch/client@0.1.7](#)

Cliente 0.1.6

23 de abril de 2026 · Cliente

Mantenimiento de la publicación.

- Cambios en la forma en que se compila y se publica el cliente. El cliente se comporta como en la versión anterior.

[@openlatch/client@0.1.6](#)