Scope · Runtime enforcement at the wire protocol layer

Runtime controls for AI agents.

Agents need production access to be useful. Static entitlements cannot govern a non-deterministic system. Hoop enforces controls on the operation itself, in flight, at the wire.

- Wire-protocol decode across PostgreSQL, MySQL, MongoDB, SSH, Kubernetes, and HTTP
- Three operation gates: allow, block, or route to a human
- Deploys behind the PAM, MCP gateway, or sandbox you already run
- Controls anchor to protocols, not to models or agent frameworks

100%
Of agent operations inspected in real time

3
Operation gates: allow, block, human in the loop

1
Enforcement point across every protocol

0
New service accounts or standing credentials

CONTENTS

> **One definition**
>
> Hoop does runtime controls for AI agents. Runtime describes data in motion: the transaction, not the row at rest. Every claim in this brief follows from that one position.

01

The access dilemma

Agents are useful in proportion to the data they reach.

Block an agent from production and it returns nothing of value. Every support chat that cannot read your records proves the point. Grant that access and you accept two new risks: service disruption and data exposure.

Regulated operators cannot absorb that risk.

A social feed can go dark for thirty minutes. A payment rail cannot. Neither can a clinical system. Regulators expect controls in place before deployment, not after the first incident.

Waiting carries its own cost. Software is a thinning moat. Competitors that ship agents first take the ground.

Entitlements assume a catalog you cannot keep.

Today's controls run on entitlements: the policy-defined model of access. An entitlement names a door. This application connects to this table. Someone has to know the door exists, then declare it.

That model needs a current inventory of every application, table, and column. Cloud estates hold thousands of tables and add more every day. The catalog goes stale the week you finish it.

> **The non-determinism problem**
>
> Agents do not repeat themselves. The industry already faced this with humans, who are also non-deterministic, and the answer was to revoke direct access. The largest breaches and outages trace back to human error, so the fix was to remove the human and route work through automation. You cannot apply that fix to agents. An agent with no path to the system does no work.

02

Architecture at the wire

Hoop runs as a lightweight proxy next to the workload, inside your network. Every connection passes through it before reaching a resource. The proxy decodes the wire protocol, so it reads the operation, the target tables, and the response rows.

Identities

AI agents
Coding, analytics, and business assistants

Engineers
Existing CLI and native clients

Support teams
Runbooks and scoped actions

Analysts
Query access to replicas

↓

Runtime control layer · wire protocol decode

Decode
Statement, target, payload, response rows

Evaluate
Behavioral rules, not per-resource entitlements

Gate
Allow, block, or route to a human

Record
Operation logged to the named identity

Self-hosted in your VPC · Outbound connections only · No data leaves your network

↓

Infrastructure

Databases
PostgreSQL, MySQL, MongoDB, warehouses

Kubernetes
kubectl sessions and pod exec

Servers
SSH and shell access

APIs and tools
HTTP services and MCP tool calls

> ### Why the endpoint is not enough
>
> One HTTP call can read data or delete it. A GraphQL service exposes a single endpoint for both. An entitlement that permits the endpoint permits both operations, because the two requests look identical apart from the payload. Control has to live where the difference lives.

03

Runtime controls

Real-time inference changed the economics of inspection. Fast language models and GPU-backed analysis make it viable to evaluate every operation as it passes, instead of pre-declaring which resources an identity may touch.

THREE OPERATION GATES

GATE 01

Allow

The operation matches expected behavior. It executes and records.

GATE 02

Block

The operation matches a prohibited pattern. It stops before reaching the resource.

GATE 03

Human in the loop

The operation carries risk. A reviewer decides, and the agent resumes on approval.

ENFORCEMENT IN FLIGHT

01 Live data masking, without a schema catalog

Hoop redacts PII and PHI in the result before it reaches a terminal, a notebook, or a model context. Detection runs through Google Cloud DLP or Microsoft Presidio. The proxy inspects output in transit, so it needs no advance knowledge of your schema and no policy per column.

02 Guardrails that stop the operation, not the connection

Destructive and mass-modification patterns stop at the gateway. Enforcement combines pattern validation on inputs, read-only connection scoping, and approval requirements on write paths. A hallucinated destructive statement never reaches the target system.

03 Approvals that do not kill the agent flow

The gateway holds the operation and routes a request to Slack or Teams. The reviewer sees the exact statement and approves in one click. The agent resumes the original query. Attribution stays on the human behind the request.

04 Recording at the operation level

Every action records against the named identity: the statement, the target, the result, and the timestamp. Logs are append-only and hash-chained. Events push to your SIEM. Auditors run a search instead of reconstructing a session.

One identity model for humans and agents

An agent carries the identity of the human who authenticated it. If an engineer starts an agent session, the agent reaches only what that engineer can reach, and every operation logs against that person.

You write no new IAM policy per agent. That approach collapses once a team runs dozens of services.

FEDERATION

OIDC against your IdP

Okta, Microsoft Entra ID, Google, Auth0, and any OIDC provider. A short-lived token carries the named user through every operation, approval, and log entry.

INHERITANCE

Group claims drive access

Existing RBAC roles carry over. No per-agent setup, no duplicated policy, no standing service account with broader reach than the user behind it.

SESSION OWNERSHIP

Sessions belong to people

When a different user re-prompts a session, Hoop requires re-authentication. Their claims then decide what the agent can see.

> **Without runtime controls**
>
> The common shortcut is a shared service account with wide read access, wrapped in prompt instructions telling the agent what to avoid. Prompt instructions are not controls. The account outlives the project, the access never narrows, and the audit trail names a machine instead of a person.

OBSERVED DEPLOYMENT PATTERNS

Global hedge fund

Mature sandboxing, authorization, and credential injection already in place. Added operation-level control for cases the existing stack could not distinguish.

Cross-border fintech

Retained existing MCP infrastructure. Layered runtime enforcement behind it for agent access to customer data.

Large insurer

Same pattern. Runtime controls added behind the current agent tooling rather than replacing it.

04

What Hoop is and is not

PAM gateways, MCP gateways, and AI sandboxes decide who connects. Hoop decides what the operation does once it is in flight. Those are different layers, and they compose.

LAYER	WHAT IT DECIDES	RELATIONSHIP TO HOOP
PAM gateway	Who connects, and who holds the credential	Keep it. Hoop deploys behind it and governs the operations that follow the connection.
MCP gateway	Which agent may call which tool	Keep it. Hoop governs what the tool call actually does against the resource.
AI sandbox	Where agent code executes	Keep it. Hoop governs what that code reaches outside the sandbox.
Runtime control layer	What the operation does, in flight	This is the layer Hoop owns. Existing tools cannot see inside the connection.

> **Hoop is not a PAM or MCP replacement**
>
> Hoop is not trying to be the best MCP gateway on the market. Teams deploy the proxy behind the access stack they already run and keep their existing single sign-on, credential custody, and tool registry. Runtime enforcement is the layer those tools do not provide.

The wire holds still.

Models change. Frameworks change. Agent protocols shift inside a quarter. PostgreSQL, SSH, and HTTP do not. Controls anchored to the wire survive the churn above them.

Buyers ask what happens when the stack turns over again. Enforcement at the protocol layer answers that question directly.

TURNS OVER IN MONTHS

Model providers. Agent frameworks. Tool-calling standards. Orchestration layers.

HOLDS FOR YEARS

PostgreSQL and MySQL wire protocols. SSH. HTTP. Kubernetes APIs.

05

Compliance evidence

Hoop generates audit evidence from every governed session. Your team assembles nothing by hand before an audit.

HOOP HOLDS

SOC 2 Type II. GDPR alignment. CNCF membership. The gateway is MIT licensed and open source at github.com/hoophq/hoop.

HOOP GENERATES EVIDENCE FOR

Your program under HIPAA, PCI DSS, NIST 800-53, and similar frameworks. Hoop does not certify your program and does not hold those certifications on your behalf.

FRAMEWORK	CONTROLS IN SCOPE	EVIDENCE EACH SESSION GENERATES
SOC 2	CC6 CC7 CC8	Append-only access log with hash chaining. Approval records per operation. Anomalous activity pushed to SIEM.
GDPR	Art. 25 Art. 30 Art. 32	PII redaction by default. Processing activity records. Cross-border access trail per named identity.
HIPAA	164.312(b) 164.312(d)	Session records with PHI redacted in transit. Identity attribution on every operation touching PHI.
PCI DSS 4.0	Req 7 Req 8 Req 10	Operation-level session records. Least-privilege enforcement per request. Cardholder data masked in transit.
NIST 800-53	AC-2 AC-17 AU-2 AU-12	Account management records. Remote access logs. Auditable events captured at the operation level.

> **Why auditors accept the same trail for agents**
>
> Agent sessions record through the same pipeline as human sessions, against the same named identities, in the same format. Teams under active audit add agents without opening a new control conversation.

06

Deployment and pilot

Hoop runs inside your infrastructure. The gateway deploys as a container in an existing cluster and needs no inbound firewall rules.

STEP 01

Deploy the gateway

One command via Helm or Docker Compose. PostgreSQL stores sessions and audit data. You bring your own managed instance.

STEP 02

Connect your IdP

Okta, Microsoft Entra ID, Google, Auth0, or any OIDC provider. Existing roles and group memberships carry over.

STEP 03

Register resources

Point Hoop at databases, clusters, SSH hosts, and internal APIs. The agent opens outbound connections only.

> **No agent install. No VPN. No new interface.**
>
> Engineers connect through psql, mysql, kubectl, or any native client. The connection string points at Hoop instead of the resource. Nothing changes for the engineer. Every session becomes governed and searchable for the security team.

A WORKING PILOT IN FOUR WEEKS

WEEK	MILESTONE	OUTCOME
WK 01	Federation enabled	Hoop trusts your IdP. Agents pre-registered. Token and group claim format validated with security.
WK 02	First connection	One database onboarded. Masking active. One approval rule live. First attributed query on a replica.
WK 03	End-to-end pilot	A real workflow runs through Hoop. Masking, guardrails, and gates tuned to what the agent actually attempts.
WK 04	Expand	Second connection. Second agent. Second team. The same controls carry over with no new policy work.

SUMMARY

Controls belong where the operation happens.

Entitlements govern doors. Agents do not use doors predictably. Hoop decodes the wire, evaluates the operation in flight, and allows it, blocks it, or hands it to a human. One enforcement point, every protocol, every identity.

That layer sits behind the access stack you already run. It anchors to protocols that outlast the models and frameworks above them.

READ

Documentation and architecture guides at hoop.dev/docs

RUN

Open source under MIT at github.com/hoophq/hoop

TALK

Technical walkthrough at hoop.dev/meet