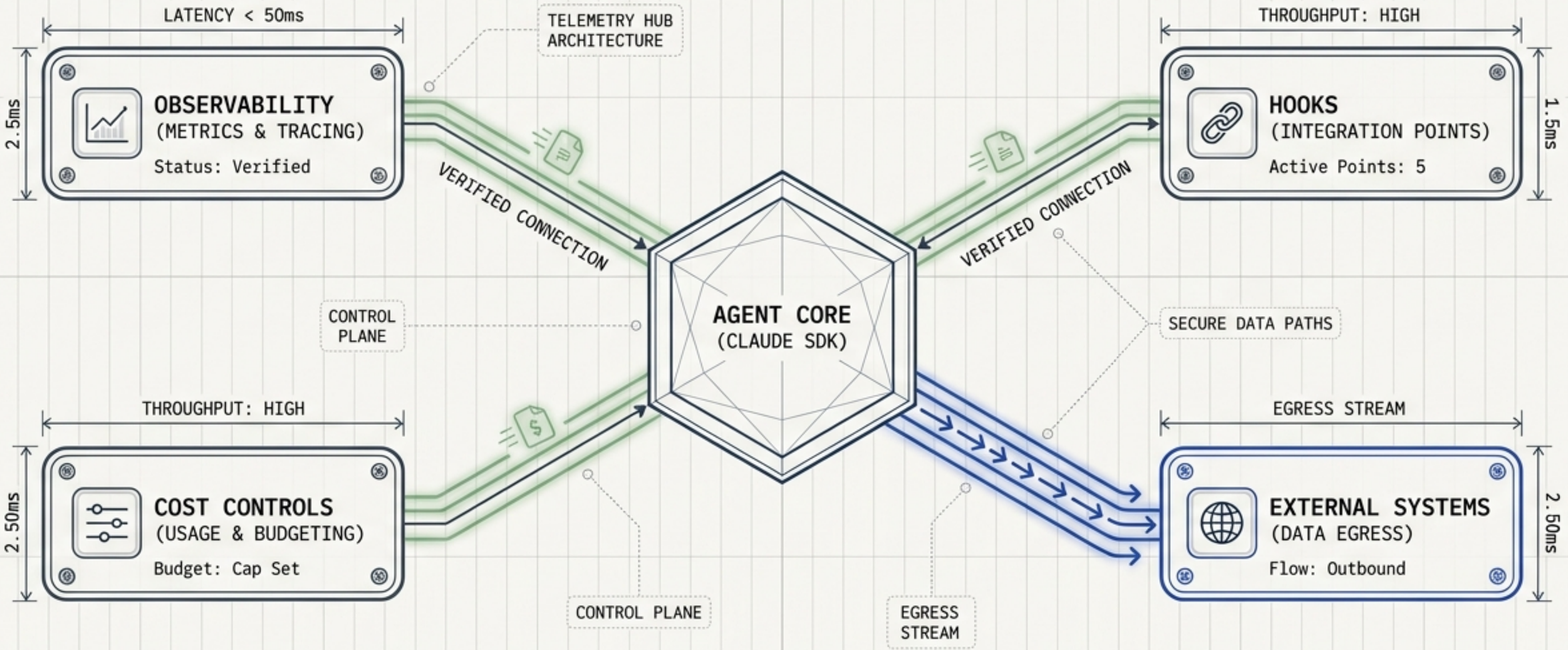
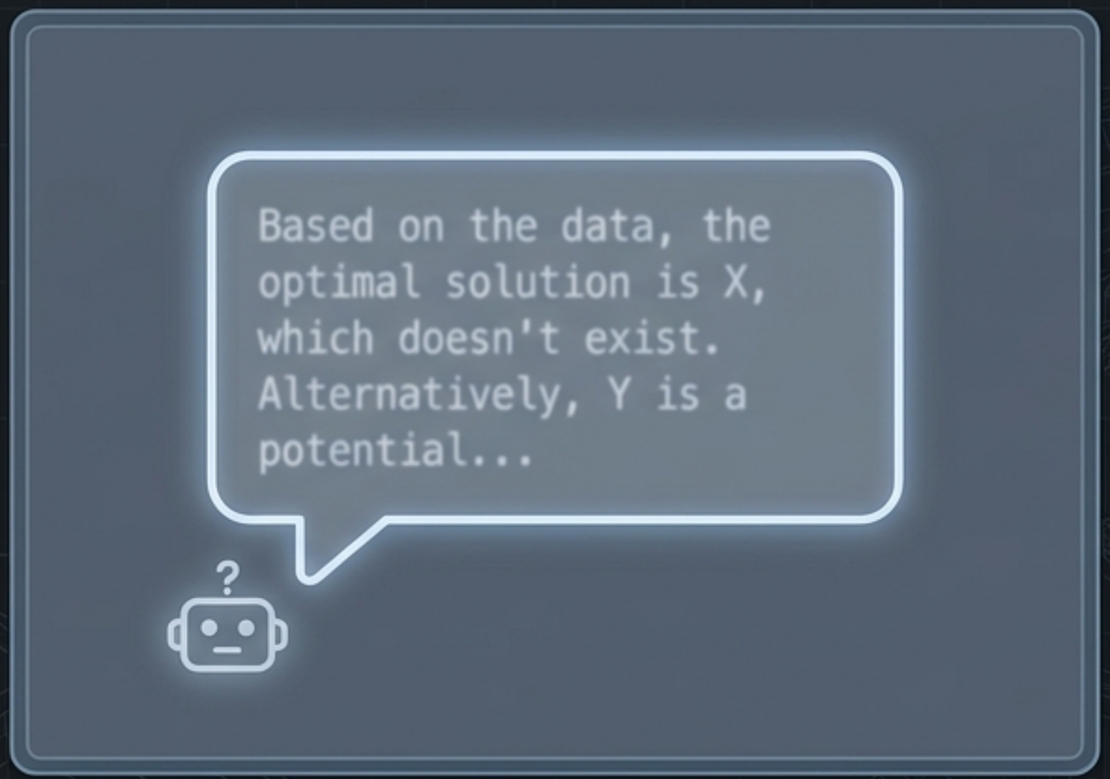
PRODUCTION READINESS PLAYBOOK

`deploying the claude agent sdk with observability, hooks, and cost controls`



The real production threat isn’t hallucination.

The Researcher’s Fear

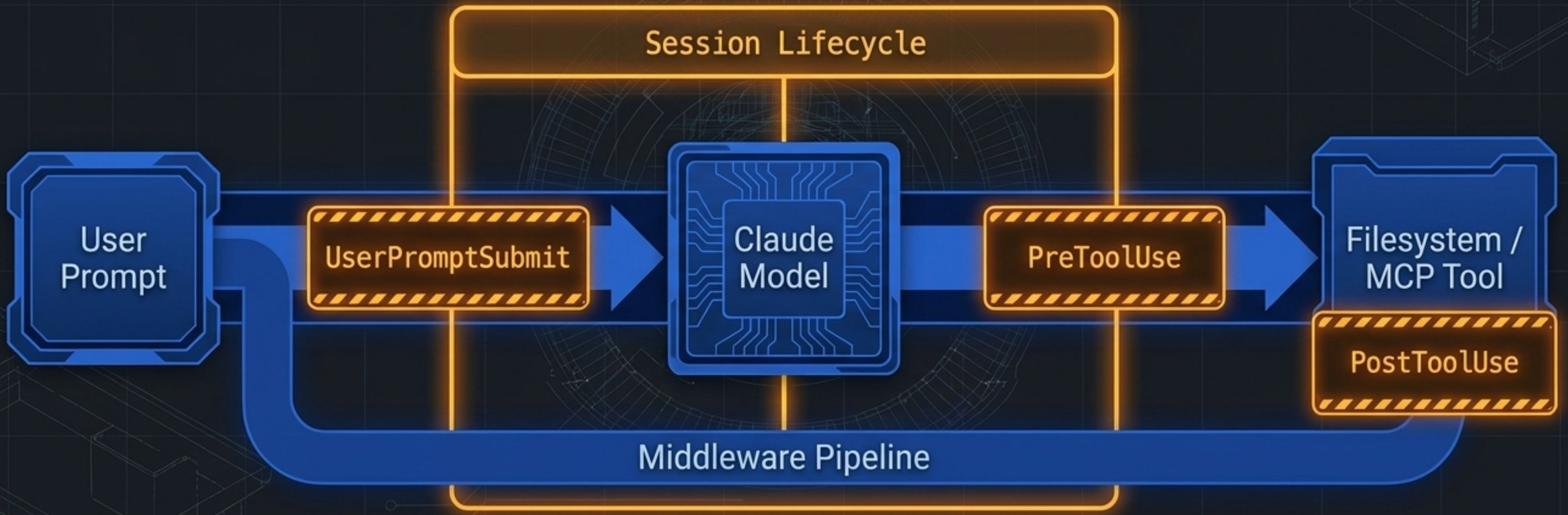


The Production Reality

It is runaway API costs and silent, destructive system modifications. In production, safety requires intercepting actions before they hit the filesystem or the billing cycle.

The Agent Lifecycle Intercept Pipeline

The SDK's hook system operates exactly like HTTP middleware—allowing you to attach synchronous functions to intercept, modify, or block actions at four critical junctions.



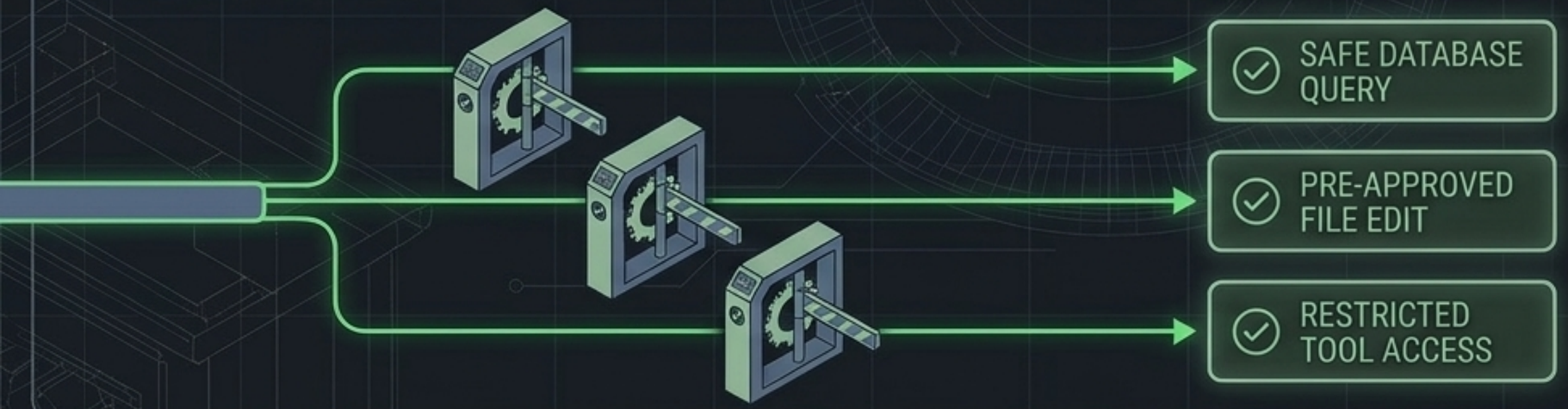
The Blast Radius: Scoping Permissions

`permissionMode: "bypassPermissions"`

DANGER

Disables ALL safety checks. Exposes destructive Bash commands and unprompted file edits. Never use in production.

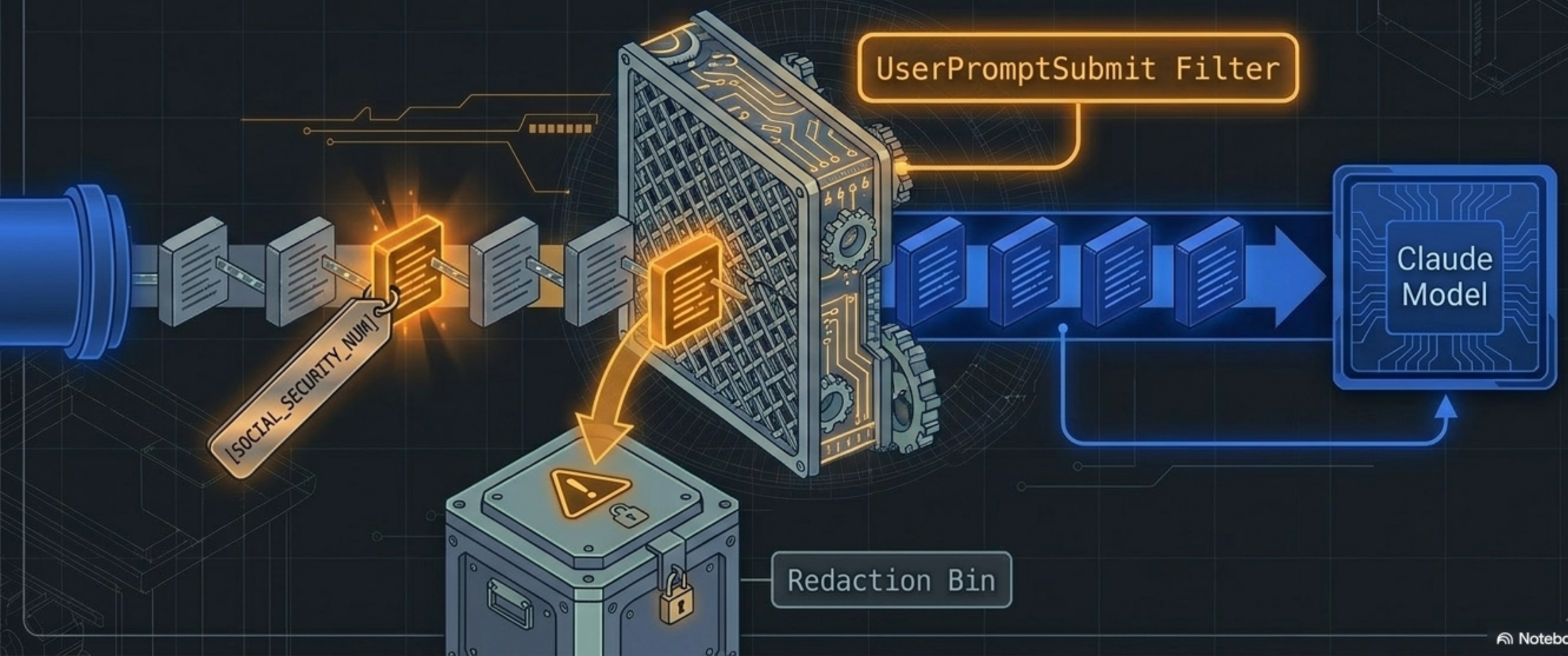
SAFETY & CONTROL `allowedTools + permissionMode: "acceptEdits"`



The safe alternative. Explicit tool grants combined with pre-approved, non-prompting file edits.

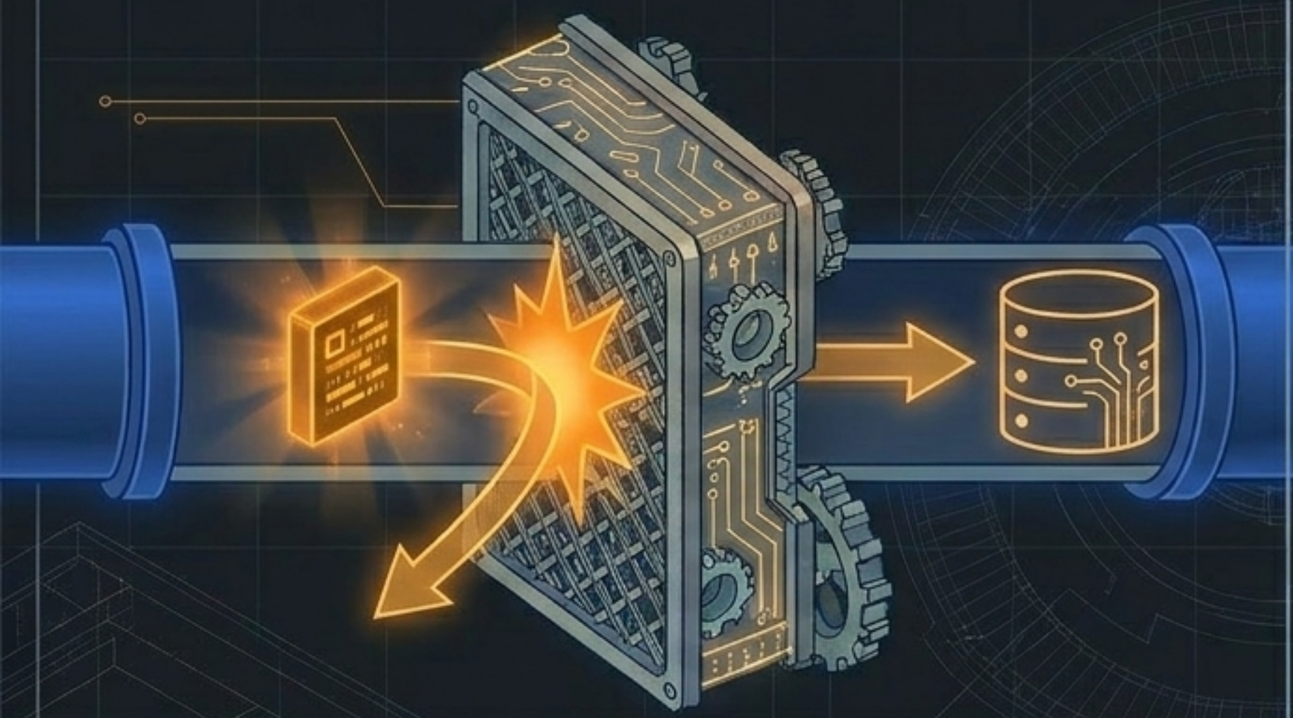
Initial Defense: Prompt Sanitization

The UserPromptSubmit hook fires the moment a message is submitted. Use it to automatically strip PII, secrets, or dangerous prompt injection patterns before they consume tokens or reach the model.



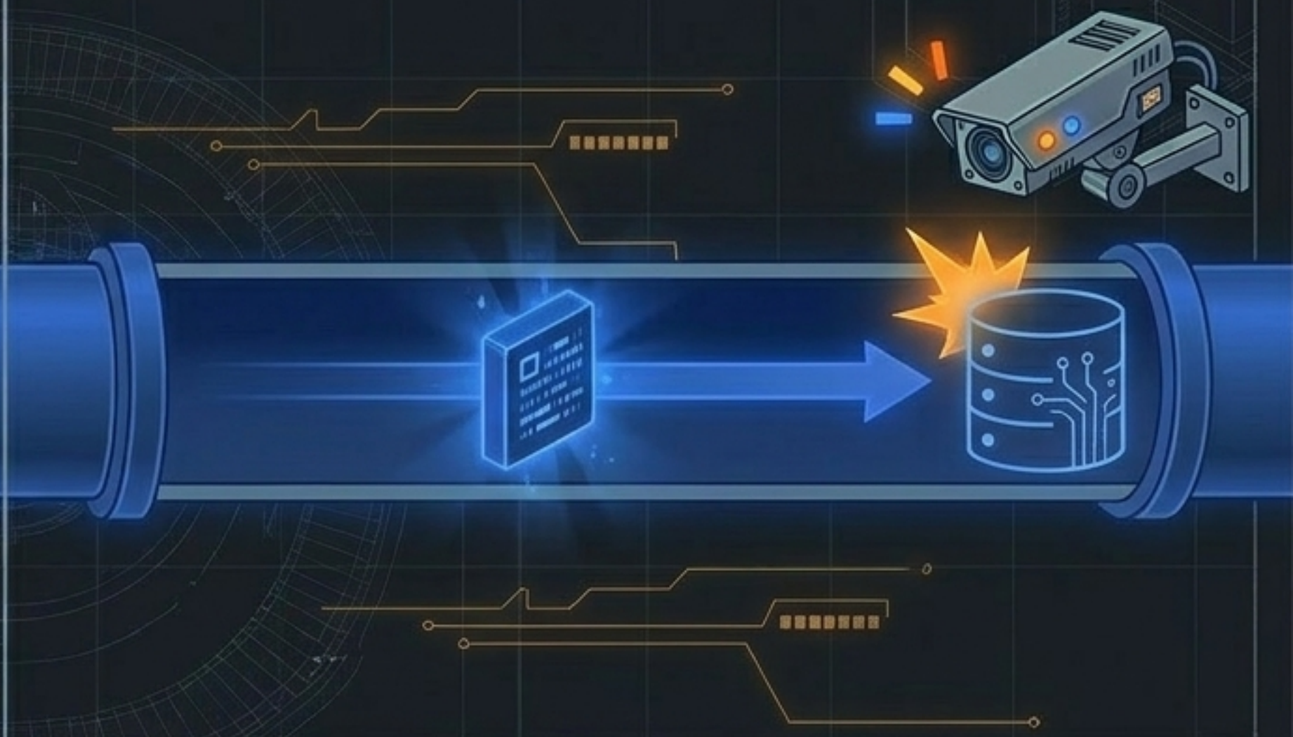
Prevention vs. Observation

`PreToolUse`



Blocks side effects. The only way to stop runaway costs or unauthorized writes before they happen.

`PostToolUse`



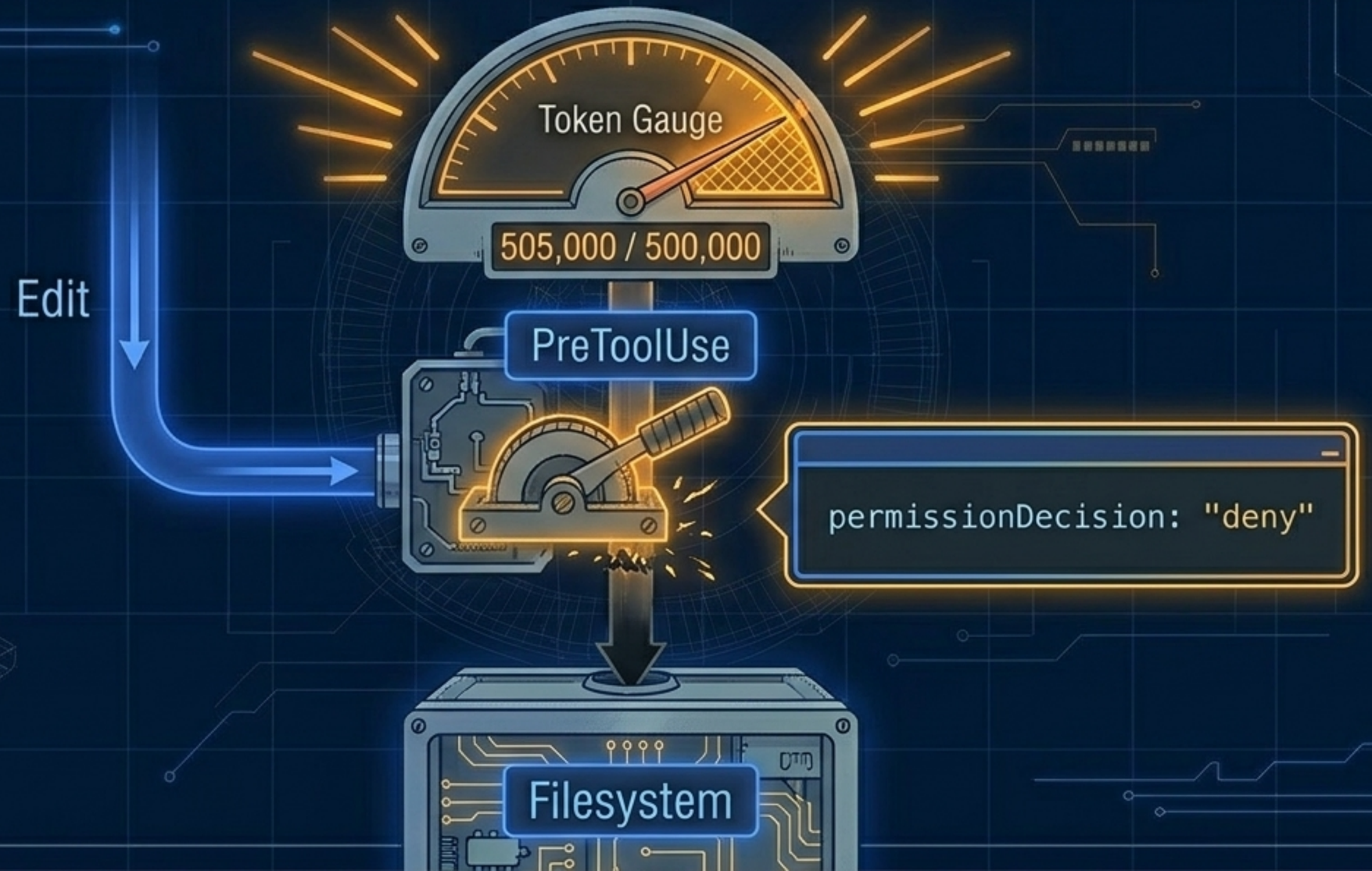
Logs side effects. Excellent for audit trails, but completely useless for preventing the action that just occurred.

The Hook Decision Matrix

	PreToolUse	PostToolUse
Execution Timing	Before tool execution	After tool execution
Primary Use Case	Cost circuit breakers, input validation	Audit logging, usage tracking
Can block side effects?	Yes	No
What it receives	Pending tool input	Tool input + execution result

Implementing the Cost Circuit Breaker

When application-managed token counters exceed their budget, the PreToolUse hook returns a deny decision. The pending tool call is blocked before execution, and the model receives feedback explaining the budget breach.



The Audit Trail: File Modification Tracking

Every successful Edit or Write operation must be logged. PostToolUse captures the exact target, result status, and session context to ensure no file is ever modified silently.

The Telemetry Divide: Session Lifecycle

TypeScript (Native Hooks)

The TypeScript SDK provides native callback hooks for session start and end events.

SessionStart

SessionEnd

Python (Manual Emission)

ClaudeSDKClient
.receive_response()

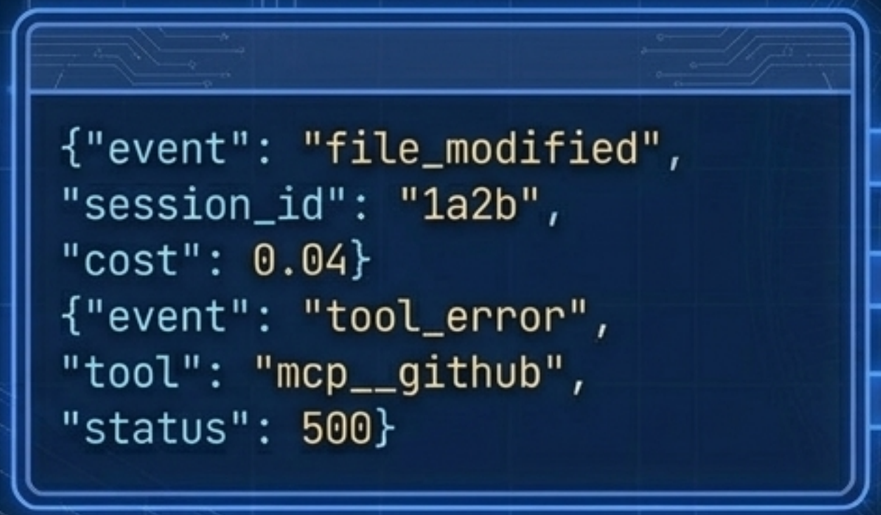
Manual Start Event

Python SDK callbacks lack native session hooks. For SDK-local telemetry, you must emit your session-start event manually when the first response arrives.

SessionEnd

Structured Logging Powers Observability

Print statements do not scale. Emitting structured JSON for every tool call allows backends like Langfuse to instantly correlate log spans onto a trace timeline, transforming raw text into actionable fleet metrics.

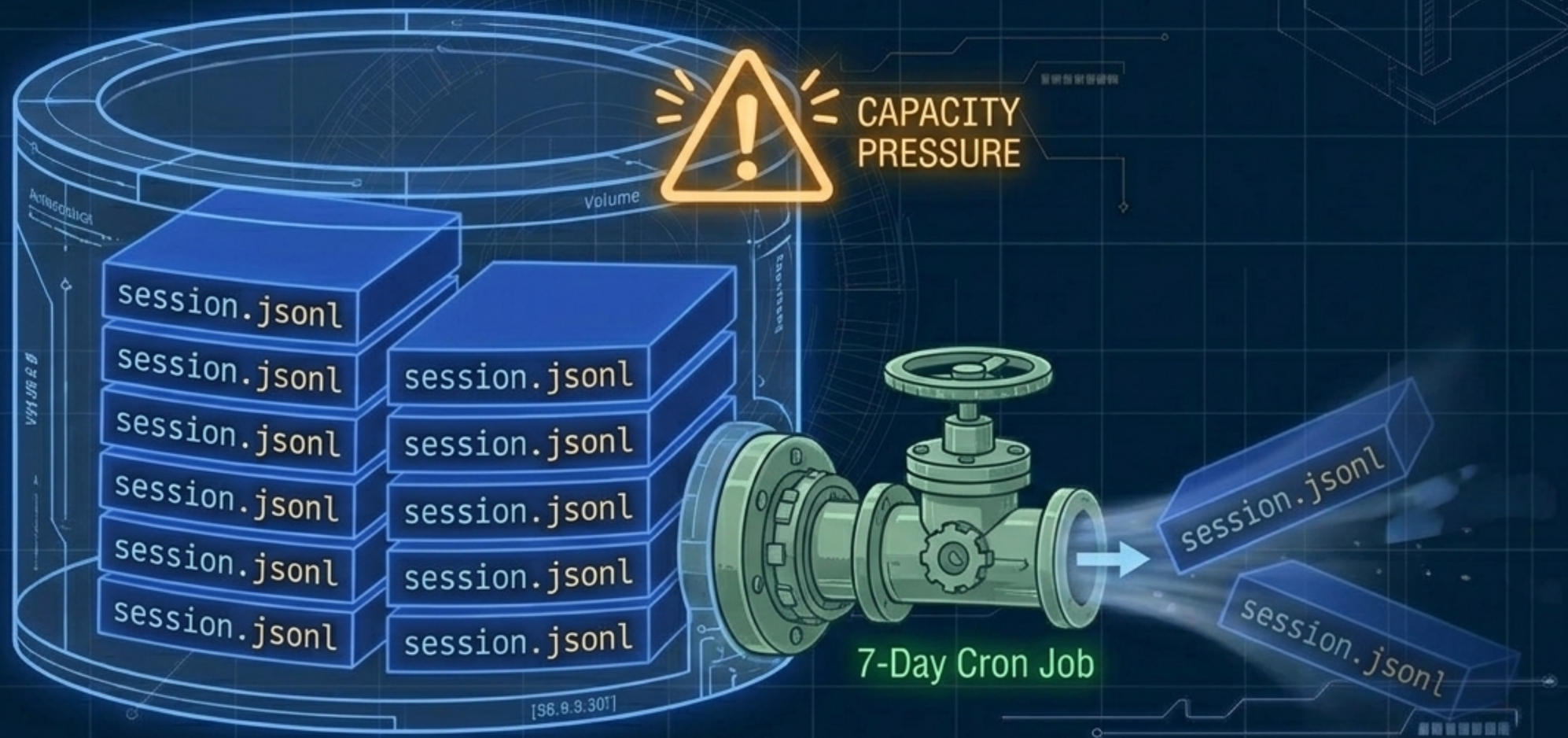


Target Environments Matrix

Lambda / Cloud Functions	Long-running Containers (EC2, K8s)	Managed Agents
Session < 15 mins	Session Unbounded	Session Persistent
Persistence Ephemeral `/tmp`	Persistence Container Disk	Persistence Anthropic Server-Side
Best For Short tasks	Best For Multi-turn, resumable tasks	Best For Async, zero-infra orchestration
Risk Cold starts wipe un-serialized sessions	Risk Unbounded JSONL disk growth	

Managing State: The Container Risk

By default, session JSONL files are written to `~/.claude/sessions/`. In long-running containers, this leads to unbounded disk growth. Always configure `CLAUDE_SESSIONS_DIR` and implement an automated 7-day log rotation cron job to prevent disk exhaustion.



Identity & Secrets Management

Never commit keys in configuration files or headers. Use explicit environment variable references so that sensitive credentials remain strictly out of version control.

ANTI-PATTERN (Hardcoded Keys)

.mcp.json

```
1 {
2   "github_token": "ghp_12345ABCD..."
3 }
```

BEST PRACTICE (Environment Variables)

.mcp.json

```
1 {
2   "github_token": "${GITHUB_TOKEN}"
3 }
```

Environment Vault

The 5-Step Pre-Flight Checklist

Minimal Permissions
No `.*` wildcards; explicit `allowedTools` & `acceptEdits`.

Cost Controls Wired
`PreToolUse` token circuit breakers active.

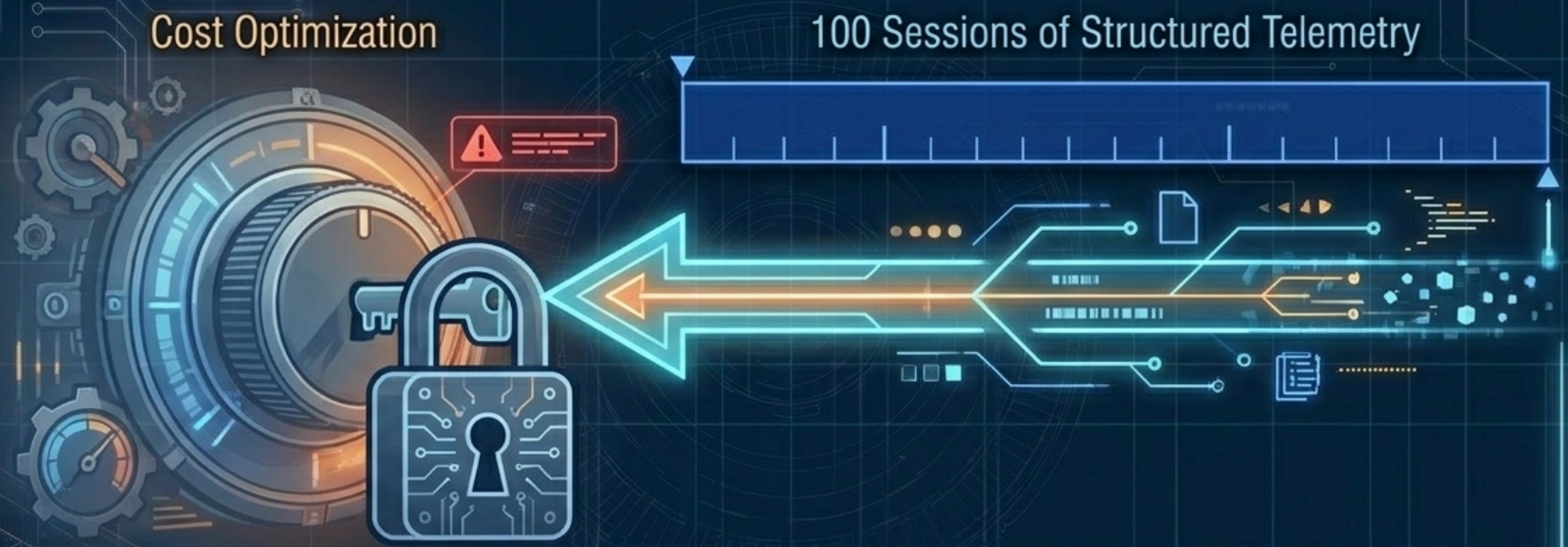
Audit Logs Active
Structured JSON capture for every `Edit` and `Write`.

Secrets Externalized
``${VAR}`` syntax utilized; zero hardcoded tokens.

Retention Policy
`CLAUDE_SESSIONS_DIR` configured with log rotation.

The Contrarian Truth: Log Before You Optimize

The immediate instinct after deployment is to tune models and cache prompts. Resist it. Until you have 100 sessions of structured telemetry, you are guessing.



The production hook stack isn't just a safety net—it is the prerequisite infrastructure that generates the evidence needed for actual cost optimization.