ANVILOGIC™

AN EVALUATION GUIDE

Build or Buy

Evaluating Agentic AI for the Security Operations Center

An evaluation guide for the security engineers, architects, and SOC leaders deciding whether to build or buy

THE SIX CAPABILITIES YOU'RE DECIDING WHO OWNS

- Agent runtime and orchestration 01
- Governance 02
- Data access 03
- Security context 04
- Repeatable workflows 05
- Evaluation and model management 06

INSIDE

01 What you're actually evaluating

02 Why the prototype convinces, and where it stops

03 The six capabilities an Agentic SecOps system needs

04 What it costs to build

05 When building makes sense

06 Key questions to ask before you decide

07 How Anvilogic approaches it

08 The bottom line

START HERE

The TL;DR

Most security teams can now build a working AI agent in a few days. A frontier model, a handful of tool calls, and a good prompt will triage a live alert in a demo, and that demo is real. The harder question is what it takes to get from that prototype to a system your whole SOC relies on across every shift, and whether your team should be the one to build it.

This guide is meant to help you answer that question for your own environment. It covers what you're actually choosing between, the six capabilities an agentic system needs before it can be trusted with real alerts, what building them costs, and the conditions under which building is the right call. It closes with twelve questions to put to your internal team and to any vendor you're considering.

THREE THINGS ARE WORTH KNOWING UP FRONT

THE MODEL 01

Both paths pay for model usage, so the real difference is the engineering and operations time it takes to run everything around the model.

THE PROTOTYPE 02

The prototype is a small fraction of the work, and most of the cost sits in the runtime, governance, and security context around the model.

THE BUILD 03

Building is a reasonable choice for a narrow set of environments, but its cost grows quickly as the scope, number of users, and the agent's authority to act increase.

40%+

OF AGENTIC AI PROJECTS CANCELED BY THE END OF 2027

Gartner predicts that more than 40 percent of agentic AI projects will be canceled by the end of 2027, and names the causes as escalating costs, unclear business value, and inadequate risk controls. Most of those projects will look fine right up until they meet production.

Source: Gartner, "Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027," press release, June 25, 2025.

What you're actually evaluating

"Build versus buy" suggests two different things are being purchased, but that isn't the case here. You're paying for model usage either way, so what actually separates the two paths is the engineering and operations time it takes to build and run the operating system around that model: the runtime, the permissions, the connectors, the testing, and the knowledge of your environment that lets an agent reason about your SOC instead of a blank context window.

Build

Your use cases and workflows

- Agent runtime
- Governance
- Data access
- Security context
- Workflows
- Evaluation

Your team builds, secures, tests, and staffs it

Buy

Your use cases and workflows

- Agent runtime
- Governance
- Data access
- Security context
- Workflows
- Evaluation

The platform provides it and keeps it current

The model layer • You pay for model usage either way

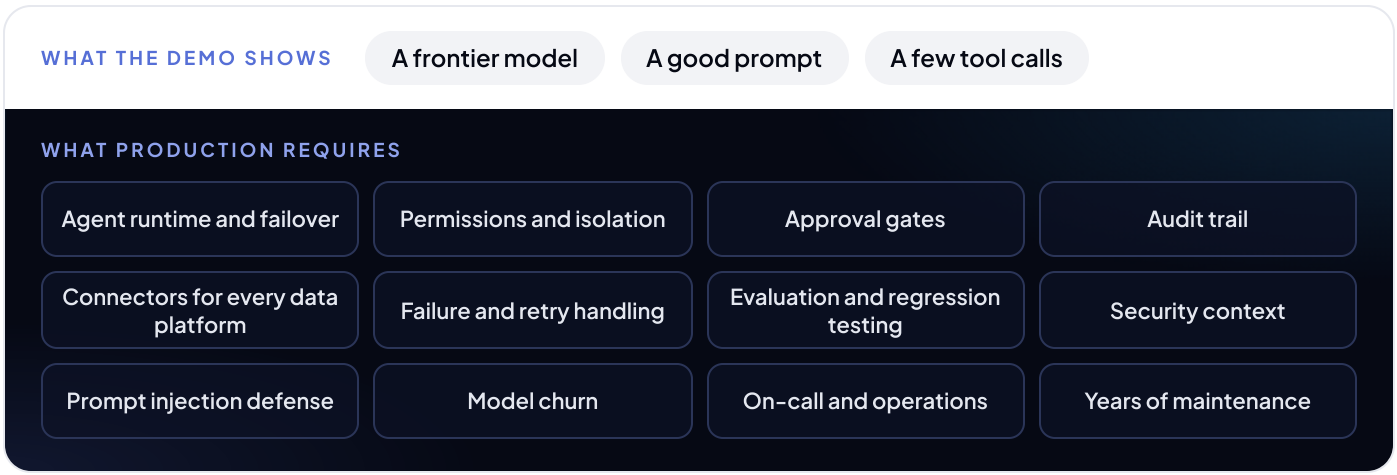
	Build on a foundation model	Buy an Agentic SecOps platform
Model usage	You pay the model provider directly	Usually metered through the platform
Agent runtime: sessions, orchestration, retries, failover	You build and operate it	Included
Permissions, audit trail, approval gates	You build and operate them	Included
Connectors to your SIEM, data lake, EDR, and threat intel tools	You build and maintain each one	Included and maintained by the vendor
Testing against every model release	You build it and staff it	Included
Knowledge of your environment	Starts empty	Varies by platform, so ask what it knows on day one
On call when it fails at 2 a.m.	Your team	The vendor

That's why comparing a platform quote to a model API bill doesn't work. One is a finished system, and the other is a single line item in the cost of building one.

Why the prototype convinces, and where it stops

There are good reasons security teams can build agent prototypes now. Natural language has become the interface, frontier models handle the reasoning, and MCP has made connecting a tool to a model close to a copy and paste job. That capability is real and belongs in analysts' hands, and a prototype that works is good evidence that a use case is worth pursuing.

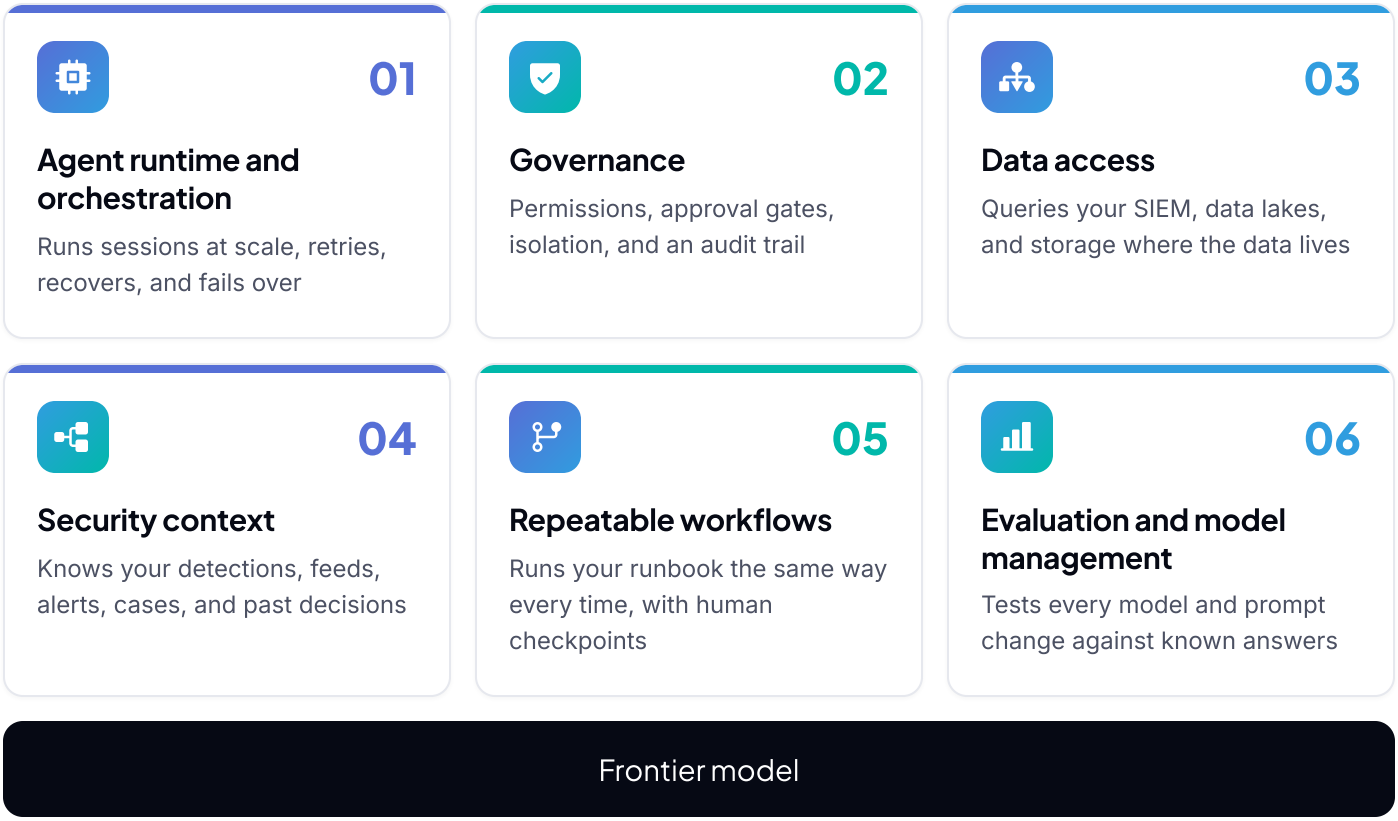
What a prototype doesn't show is how much changes between the demo and production. The model stays the same, but almost everything around it has to be rebuilt to handle more users, more data, a whole team building and changing it under version control, failures nobody's watching, and the scrutiny of an audit.



	THE PROTOTYPE		THE PRODUCTION SYSTEM
Users	One person who wrote it	→	Every analyst, every shift, multiple business units
Sessions	Can run several, but hard to share, resume, or cancel	→	Hundreds in parallel, shared, resumable, and cancellable
Data	One clean alert	→	Thousands of alerts a day across a SIEM, a data lake, and vendor feeds (some of them malformed)
Failure	Someone is watching and reruns it	→	Nobody is watching, so it has to retry, degrade safely, and log what happened
Permissions	The builder's credentials	→	Every action bounded by the launching analyst's access
Audit	A terminal scrollback	→	Every tool call, decision, and reasoning step stored and reviewable
Context	Whatever was pasted into the prompt	→	What fired, what was tuned, what was decided before, and why
Testing	"It worked when I ran it"	→	A regression suite that runs against every model update and every prompt change
Security	None	→	Prompt injection defense, supply chain hygiene, secrets handling, and memory governance
Ownership	A side project	→	A system someone has to maintain for years, which still breaks over time as vendor schemas, patches, servers, and models change

The six capabilities an Agentic SecOps system needs

Whether you build or buy, an agent that works real alerts needs six capabilities underneath it. The list is the same either way, which makes it the most useful way to compare an internal plan against a vendor's platform.



CAPABILITY 01

Agent runtime and orchestration

WHAT IT IS

The runtime is what actually runs agent sessions. It starts them, coordinates tool calls, keeps state between steps, retries when something fails, and lets analysts resume or cancel work. In a prototype, that's a script on someone's laptop; in production, it's handling hundreds of sessions in parallel for analysts across every shift.

WHY IT MATTERS IN A SOC

An agent that closes or escalates alerts is part of your detection and response path, so it needs the same operational rigor as the SIEM it reads from. The failures are specific: a triage run that hangs until the alert ages out of its SLA, a model API that goes down during an active incident, a SIEM that returns a partial result set the agent treats as complete, and a full shift of analysts stalling on rate limits at shift change. Each one has to be handled before you trust the system with a real alert.

WHAT GOOD LOOKS LIKE

A production runtime has to absorb each of these on its own, without anyone watching, so it's worth asking how an internal plan or a vendor handles every one:

- **Concurrency.** Hundreds of sessions run in parallel for analysts across every shift, and each one can be shared, resumed, or cancelled.
- **Timeouts.** When a triage run hangs mid-investigation, the runtime limits how long it can run, saves checkpoints, and resumes, instead of letting the alert age out of its SLA.
- **Provider outages.** When the model API goes down during an active incident, work is queued or fails over and degrades safely, and the analyst can see what's happening.
- **Rate limits.** When a full shift of analysts hit the model at shift change, requests are queued and prioritized, each tenant has its own budget, and results are cached where possible so half the team doesn't stall.
- **Crash recovery.** When a worker dies three steps into an eight-step investigation, the session state is saved and the steps are safe to rerun, so it restarts from the last checkpoint instead of from scratch.
- **Storage.** Uploaded evidence, result sets, and generated files live somewhere governed, with permissions and retention, and stay tied to the session that produced them.
- **Latency.** Caching, parallel work, and context that's already assembled keep a verdict from taking four minutes, which is usually the point where analysts stop using the tool.



CAPABILITY 02

Governance: permissions, approvals, and audit

WHAT IT IS

Governance decides what an agent's allowed to do and keeps a record of what it did. That covers role-based access to every agent, workflow, and session; boundaries between business units or tenants; points where the agent pauses for a person to approve; and a trail of every tool call and reasoning step that an auditor can actually read.

WHY IT MATTERS IN A SOC

An agent running on its builder's credentials can do anything the builder can do, which is rarely what you want when a whole team shares it, so every action needs to be bound by the permissions of the analyst who launched it. Approvals for allowlisting, deployments, escalations, and case actions should follow your own process rather than the agent's defaults. The agent is also a new attack surface. A prompt injection hidden in an alert or a log line, a malicious package in the agent's dependencies, or a vulnerable library can all be turned against you, and governance can't be bolted onto an agent runtime after the fact. It has to be there on day one.

WHAT GOOD LOOKS LIKE

Whether you build or buy, governance should meet these criteria:

- **Access control covers every object.** Roles decide who can read, edit, deploy, view outcomes, and manage access for every agent, workflow, and session, and those grants can be revoked.
- **Tenant boundaries carry all the way through.** Organization, user, workflow, session, and agent identifiers travel through storage, API paths, tool calls, and code execution, so one business unit's agent can't reach another's data. A model on its own knows none of this.
- **Approvals are part of the run.** The system can pause mid-run, ask a person for structured input, record the approval or denial, and pick back up cleanly, following your own manual or automatic settings for allowlisting, deployments, escalations, and case actions.
- **The trail shows the work, not just the answer.** Reasoning, tool calls, and their results are stored with the session separately from the final conclusion, so an analyst or auditor can see what the agent actually did.
- **The AI supply chain is defended,** including prompt injection from alert content or log lines, malicious npm and PyPI packages, and vulnerable libraries across the agent's dependencies.



CAPABILITY 03

Access to data where it already lives

WHAT IT IS

Agents need to query your security data, which for most teams is spread across a SIEM, one or more data lakes, cloud storage, and vendor alert feeds, each with its own query language, performance limits, and quirks.

WHY IT MATTERS IN A SOC

An agent is only as good as the data it can reach and the completeness of what comes back. A connector that returns a partial result set leads the agent to reason about half the picture like it's the whole thing. At scale, a lot of sessions querying the same platform at once also creates contention that slows down your analysts.

WHAT GOOD LOOKS LIKE

Data access should meet these criteria:

- **Queries run in place,** in each platform's own query language, under the access of the analyst who launched the agent.
- **Results are validated,** failed queries are retried with backoff, and an incomplete result set is flagged as incomplete instead of being treated as the whole picture.
- **Contention and queueing are handled,** so many sessions querying the same SIEM or data lake don't slow your analysts down.
- **Results from different platforms come back in a consistent shape,** and large result sets are sampled and analyzed instead of being dropped or cut off.
- **Each additional SIEM or data lake is its own connector to build and maintain,** so count how many your environment needs today and how many it'll need in two years.

CAPABILITY 04

Security context

> This is the capability that's most often underestimated, and it's the one that separates an agent that gives generic answers from one that understands your SOC.

WHAT IT IS

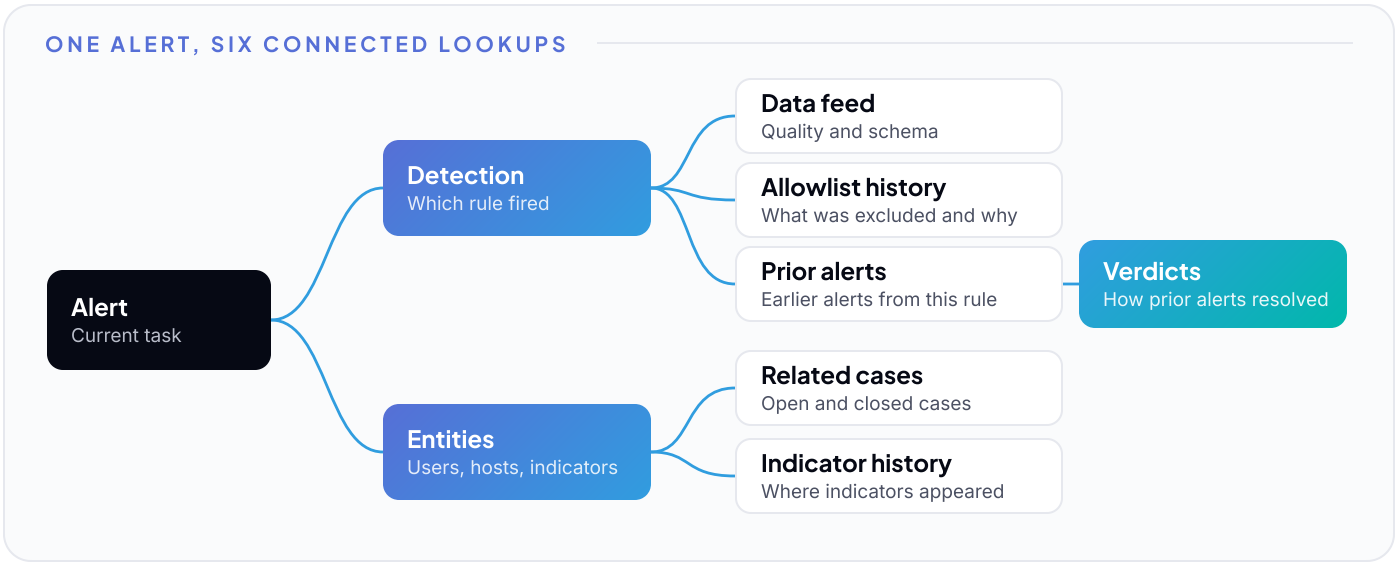
Security context is everything an agent needs to know about your environment to make a good call. A model starts every session with a blank context window. It doesn't know what data you collect or where it lives, which detections you've deployed, what alerts they've produced, which cases turned out to be malicious, what was tuned and why, or which threats your team has decided matter most this quarter.

WHY IT MATTERS IN A SOC

To assess a single alert adequately, an agent needs to know:

1. which detection fired
2. which data feed that detection reads and how reliable the feed is
3. what's been allowlisted on that detection and why
4. how earlier alerts from the same detection were resolved
5. whether the users, hosts, or indicators involved show up in any open or closed case
6. whether those indicators have been seen anywhere else in your history

That's a chain of connected questions, three to six steps long, across very different kinds of records.



That's why security context is a set of relationships instead of a pile of records. A data warehouse answers aggregate questions well, but struggles with a chain of connected lookups like this one, and a vector store finds text that resembles your question but can't tell you what depends on a given data feed. Neither one, and no prompt, gives the agent the shape of your SOC.

Decisions are where the long-term value builds up. When verdicts, tuning calls, case closures, and allowlist reasons are recorded (along with who made them and why), later sessions can draw on that judgment as worked examples, and the system's calls improve without anyone rewriting the instructions.

Memory is where generic approaches get risky.

UNINSPECTABLE
Memory that analysts can't inspect can bias every verdict.

STALE
Stale memory keeps a retired tuning rule or a decommissioned host alive.

OVER-BROAD
Over-broad memory leaks one team's context into another team's decisions.

POISONED
A malicious log line can write memory that's persistent across sessions.

WHAT GOOD LOOKS LIKE

Whether you build or buy, security context should meet these criteria:

- **It's tied to the system of record.** Context should come from the platform that onboarded the data, deployed the detection, produced the alert, and recorded the verdict. An agent sitting on top of a SIEM you don't control has no lineage to follow.
- **Facts are dated,** so you can ask what the system knew the second an alert fired.
- **Facts carry their source.** Each one records whether a person, a named agent, or a pipeline created it, so a human-approved decision counts for more than an unreviewed one.
- **Decisions are captured,** including what they were about and who made them.
- **Memory is visible and scoped.** Analysts can inspect, correct, and delete it, and it stays within the right team or business unit.
- **Agent writes go through the same controls people do,** with the same permissions, approvals, and audit trail.

To learn more about security context and governed memory work in practice, read AI agents need more than just data, they need the context of your SOC.

CAPABILITY 05

Repeatable workflows with human checkpoints

WHAT IT IS

An individual agent can reason through a task, but on its own, it runs that task the way its builder assumed a SOC works. Workflows chain agents and tasks into a process that runs the same way every time, follows your team's actual runbook, and stops for a person wherever your process needs one.

WHY IT MATTERS IN A SOC

Consistency is what lets you trust automation across multiple shifts and teams. If the same alert takes a different path depending on who launched the agent, you can't measure, audit, or improve it. Workflows also need to be visible, with every step named, ordered, and inspectable, so an analyst reviewing a result can see what happened and where a person approved it.

WHAT GOOD LOOKS LIKE

Workflows should meet these criteria:

- **Your team writes them in plain language,** and they follow your runbook rather than the agent's defaults.
- **Human checkpoints sit wherever your process needs one,** and the run pauses and resumes cleanly around them.
- **Every run is recorded,** including which alerts it ran on, which became cases, the steps taken, and the justification at each decision.
- **Later runs can draw on earlier runs** of the same workflow on similar alerts as worked examples, so its judgment improves without anyone editing its instructions.



CAPABILITY 06

Evaluation and model management

WHAT IT IS

Evaluation is the testing that tells you whether the system is still making good calls. It includes a set of known alerts with known correct verdicts, a baseline to compare against, and a regression suite that runs whenever a prompt or model changes.

WHY IT MATTERS IN A SOC

Frontier models change every few months, and each release can shift tool-calling behavior, verbosity, refusals, and verdict quality. Without evaluation, a model update can start closing real intrusions as benign, and nobody will notice until it matters. Keeping up with the latest model means re-benchmarking, adjusting prompts, and rerunning regression every cycle, which can take an internal team weeks each time.

WHAT GOOD LOOKS LIKE

Evaluation should meet these criteria:

- **There's a golden dataset** of real alerts with known verdicts, and a baseline every change is measured against.
- **Verdicts come back in a structured format,** so they can be compared run to run and drift gets caught when the same alert gets a different verdict on Tuesday than it did on Monday.
- **Model versions are pinned,** tool-calling behavior is covered by contract tests, and new models roll out in stages, because a version change can alter how the agent calls its tools.
- **The regression suite runs on every model update and every prompt change,** and the results of the last run are something you can show.

What it costs to build

The estimates below describe what a capable internal team would need to reach (rough) parity with a first production version of an Agentic SecOps system. Here's what each workstream covers and how long it typically takes:

Agent runtime and orchestration

TIME
6 to 9 months

COVERS
- Sessions
- Tool calling
- Retries
- Timeouts
- Provider failover
- Rate limiting
- Crash recovery
- Streaming
- Storage for what agents produce

Governance

TIME
4 to 6 months
then it's permanent

COVERS
- Role-based access
- Tenant isolation
- The audit trail
- Approval gates
- Prompt injection defense
- Secrets
- Supply chain hygiene

Data access

TIME
3 to 6 months
per platform family, then it's permanent

COVERS
- Connectors and query generation for each SIEM and data lake
- Contention handling
- Normalizing results at scale

Security context

TIME
12 to 18 months
to reach the point where it starts compounding

COVERS
- The data model
- Ingestion
- Entity resolution across hostnames, IPs, and users
- Dated facts
- The interface that assembles context for each session
- Writing decisions back

Evaluation

TIME
Permanently

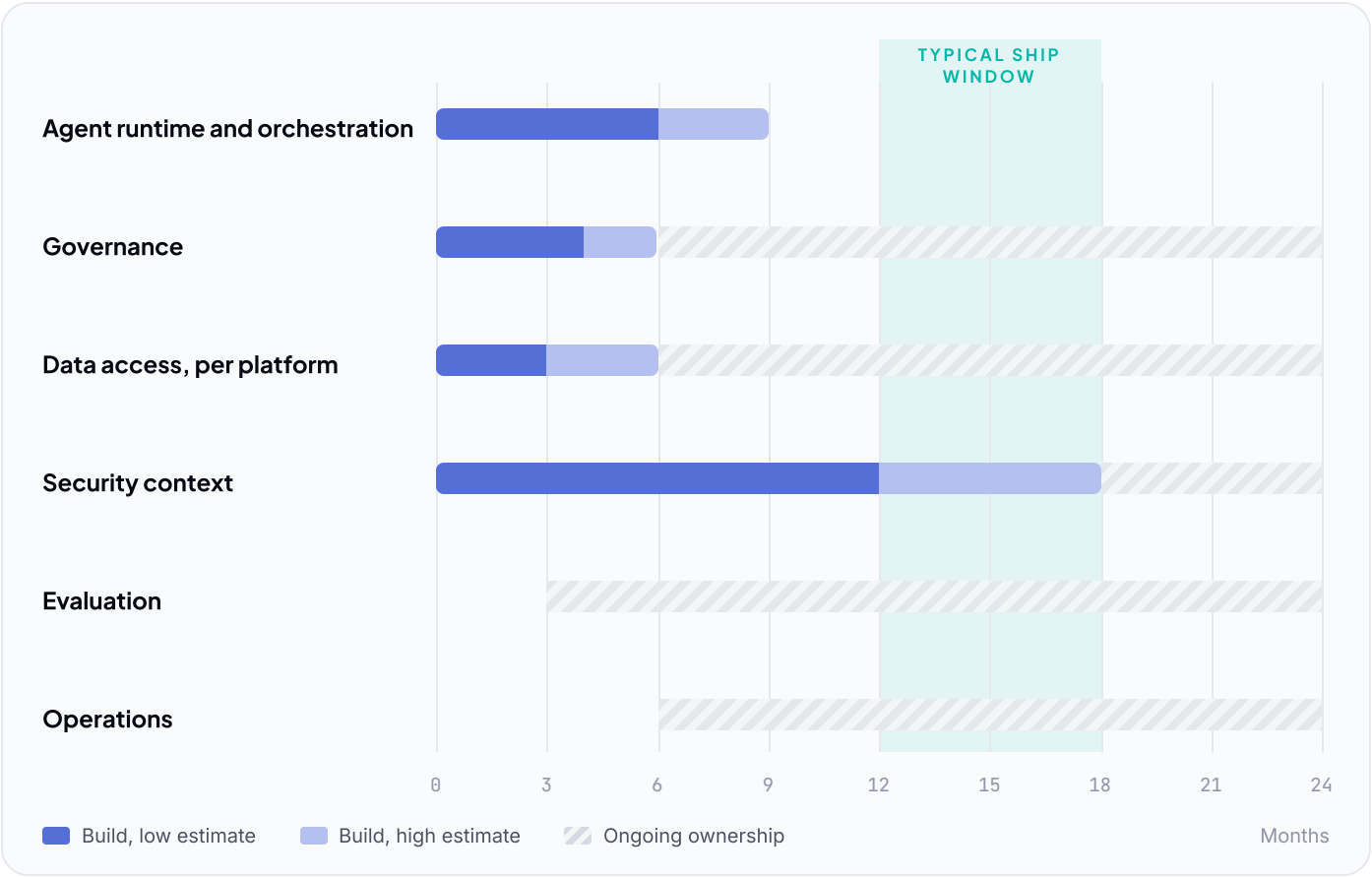
COVERS
- Golden datasets
- Verdict baselines
- Benchmarking on every model release
- Drift detection

Operations

TIME
Permanently

COVERS
- Observability
- On-call
- Deployment
- Rollback
- Cost controls

HOW THE WORKSTREAMS STACK UP OVER TIME



TO SHIP

8 to 10 engineers, 12 to 18 months

TO RUN

2 to 3 engineers, every year after

Together, that's ~8-10 engineers for 12-18 months to ship, assuming the agent only has to work with one data platform, like Splunk or a single data lake. After launch, a standing team of 2-3 is needed to keep it running, secure, and current with model releases. Both numbers can go up depending on the complexity of your environment and your requirements, like 24/7 coverage. Each additional SIEM or data lake adds its own connector work on top of that, and you'll still be paying for model usage, which is usually the only cost teams count when they compare building to buying.

THE COST THAT RARELY MAKES THE SPREADSHEET

There's also a cost that rarely makes it onto the spreadsheet: In most security organizations, the engineers available for this work were hired as detection engineers, threat hunters, and incident responders. Every quarter they spend maintaining an agent is a quarter they aren't spending on new detections, closing coverage gaps, or tuning alerts that are wearing down your analysts.

When building makes sense

Building is a legitimate choice for some environments, and a prototype is always worth building to learn what you need. The real question is whether your environment fits the conditions where a build stays manageable, and whether it will keep fitting as the use case grows.

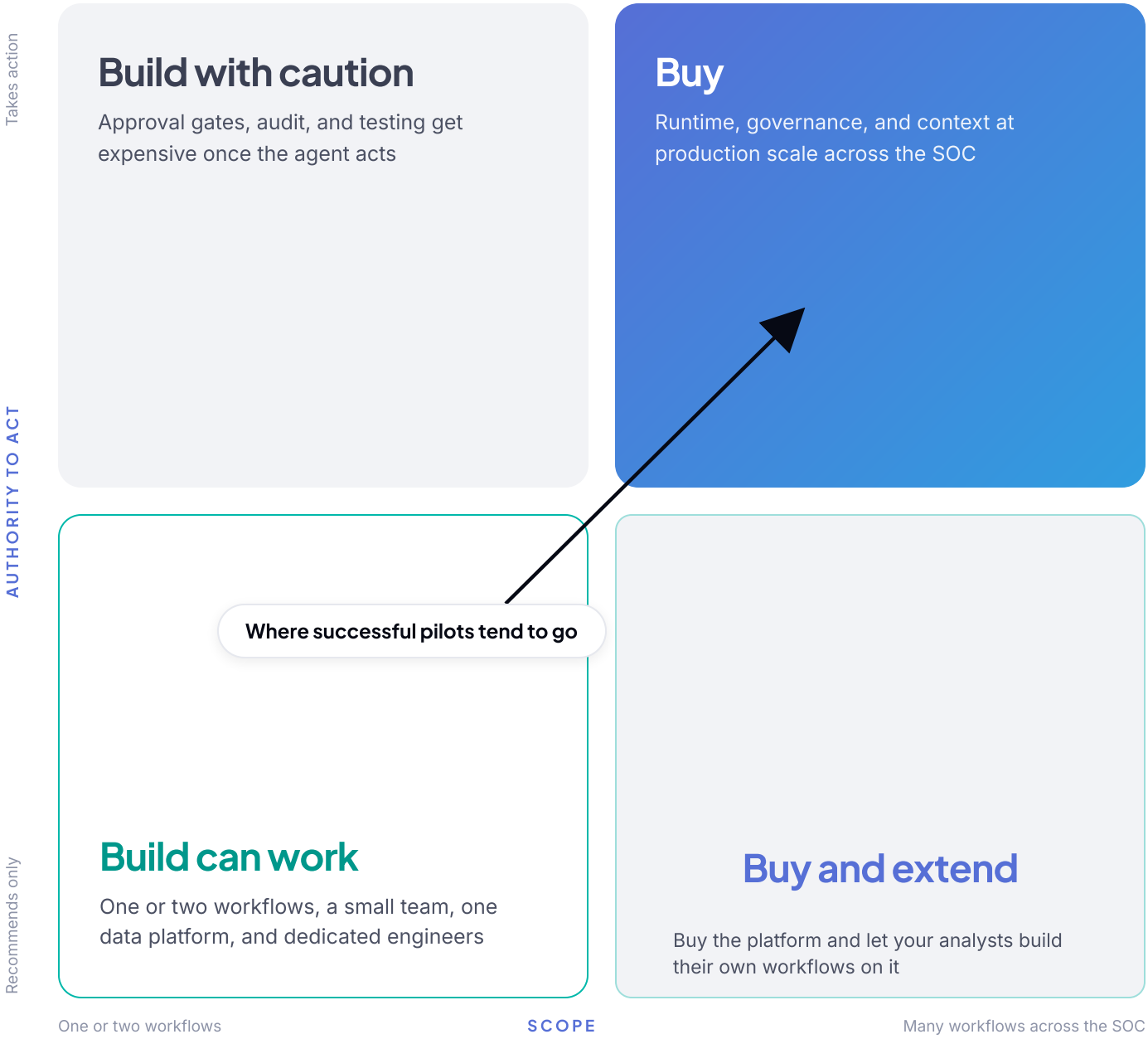
BUILDING TENDS TO WORK WHEN MOST OF THESE ARE TRUE

- **The scope is narrow.** You're automating one or two well-defined workflows, like enrichment for a single alert type or case summaries for shift handoff, rather than building a platform for the whole SOC.
- **The agent advises rather than acts.** It reads data and makes recommendations, and a person takes every action, which keeps the governance and audit burden small.
- **The user base is small.** It's a handful of analysts on one team, not several shifts or business units with different permissions.
- **You have platform engineers to spare.** The people who build and maintain it are a dedicated engineering group, not detection engineers and responders pulled away from security work.
- **Your data lives in one place you control,** so connector and contention work stay small.
- **You have requirements no vendor can meet,** like an air-gapped environment or data residency rules that rule out any external platform.
- **Occasional failures are acceptable.** The workflow doesn't need to work every time, since getting an agent to work most of the time is far easier than getting it to work 99% of the time.
- **The data isn't sensitive.** The agent never touches regulated or confidential data, so it doesn't need a full governance layer around it.

WHAT CHANGES AS IT SCALES

Each of those conditions tends to erode over time. A successful pilot picks up more users, which brings permissions, isolation, and contention work. An agent that recommends gets asked to *act*, which brings approval gates, audit, and a much higher bar for testing. A second data platform doubles the connector work, and every model release resets part of the evaluation. If you build, decide in advance which of these thresholds would trigger a second look, so the build doesn't eventually grow into a platform program nobody approved.

WHERE YOUR ENVIRONMENT LANDS



Buying and extending is where a lot of teams typically land. You buy the runtime, governance, data access, and security context, which are the expensive and permanent parts, and your team builds its own workflows on top of them in plain language, connecting its own tools where it needs to. Your analysts keep the ability to automate their own processes, and nobody on the security team ends up responsible for keeping a runtime alive.

Key questions to ask before you decide whether to build or buy

Use these with your internal team before approving a build, and with any vendor you're evaluating. Whoever you're talking to should be able to answer each one with evidence, not a description.

	QUESTION	EVIDENCE TO ASK FOR
01	If the agent breaks at 2 a.m. during an incident and no one fixes it, is that OK? If not, what's the rollback?	A named on-call rotation, a runbook, and a rollback that has been tested
02	How do we know a model update didn't change verdict behavior?	The regression suite and the results of its last run
03	Can an agent take an action the launching analyst isn't permitted to take?	A live demonstration that it can't
04	Do we need a record of every tool call and decision that an auditor can read without an engineer's help?	A session trail an auditor can open and follow on their own
05	What happens when the model provider is down or throttling us mid-investigation?	Failover behavior, queueing, and what the analyst sees on screen
06	Can the agent query our SIEM and data lake in place, under each analyst's own access, and does it know when a result set is incomplete?	A live query across two of your platforms, including how a partial result is handled
07	How does the agent know which detection fired, which feed it read, what was allowlisted on it, and why?	An agent answer traced back to its source records
08	Who reviews the memory the agent builds up, can we inspect, correct, scope, or delete it, and how is PII protected?	A memory view analysts can use, plus retention and PII handling rules
09	Who defines the workflow steps, and can the same workflow run across both our SIEM and our data lake?	A workflow your analysts wrote, with its approval points, running end to end
10	How is a prompt injection hidden in alert content or a log line handled?	The specific defenses and how they're tested
11	Who maintains this in year three, and what security work won't they be doing instead?	Named owners and a staffing plan, or for a vendor, the support model and roadmap
12	What does it cost fully loaded, including model usage?	Engineering time, infrastructure, model usage, and maintenance, not just licenses or tokens

How Anvilogic approaches it

Anvilogic is the Agentic SecOps platform. It onboards, searches, detects, and investigates on top of the stack a SOC team already has. It runs alongside a SIEM like Splunk or Microsoft Sentinel, or standalone on a security data lake like Snowflake or Databricks, so nothing gets ripped out. Here's how it covers each of the six capabilities in this guide.

Agent runtime and orchestration AI OPERATING SYSTEM	The AI Operating System coordinates agents and manages sessions, retries, failover, rate limits, and crash recovery for many analysts at once, in production today.
Governance BUILT INTO THE RUNTIME	Role-based access, tenant isolation, and approval gates are built into the runtime. Every tool call, reasoning step, and result is stored with the session, and org admins control which tools and MCP connections agents can use.
Data access FEDERATED SEARCH	Federated Search queries SIEMs, data lakes, and cloud storage in place under each analyst's own credentials, and handles contention, queueing, and large result sets.
Security context ENTERPRISE SECURITY GRAPH	The Enterprise Security Graph is built on the platform's own record of onboarding, detections, alerts, verdicts, tuning, and cases. Facts are dated and carry their source, decisions are captured, and tenants are fully isolated.
Repeatable workflows BLUEPRINTS	Blueprints chain agents into workflows your analysts define in plain language, with a human approval checkpoint wherever your process needs one. Every step is named, ordered, and inspectable.
Evaluation and model management MODEL EVALUATION	New models are benchmarked against SOC workflows before they're adopted, and a quality floor is held across releases, so your team doesn't have to run that cycle itself.

Anvilogic also fits the buy and extend path. Customers write their own Blueprints and connect their own tools through the MCP server, so teams that want to build can do it on top of the runtime, instead of underneath it.

CUSTOMERS RUNNING IT IN PRODUCTION HAVE SEEN THESE RESULTS

70+ hrs/day
SAP recovered more than 70 analyst hours per day and added 35% more detection coverage in one week, with 60 to 80% less engineering effort.
SAP

<2 min
A large financial services company triages alerts in under two minutes across thousands of events per month.
Financial services

10x
Zendesk onboards new data feeds 10 times faster, at about seven minutes per feed.
Zendesk

$1M
A Fortune 100 telecommunications company cleared a 100-feed onboarding backlog with a Blueprint that onboards a feed in about 15 minutes, saving roughly $1 million in consulting and moving eight consulting staff to other work.
Fortune 100 telecom

3x
A fintech company more than tripled the attack techniques it can detect, going from 25 to 80% coverage in nine weeks with no added headcount.
Fintech company

$1.2M
A Fortune 500 financial services company that couldn't run detections across its data silos now deploys them across Splunk and Snowflake from one place, saving $1.2M and more than 5,000 hours.
Fortune 500 finance

08

The bottom line

The prototype your team built is proof that the use case is real, and the best way to use it is as a requirements document. It shows you which workflows matter, what data the agent needs, and where your analysts want a checkpoint. From there, the decision comes down to how many of the six capabilities in this guide your team wants to own for the next several years, and whether your environment fits the narrow conditions where owning them stays manageable.

IF IT DOES

If it does, build deliberately and set the thresholds that'll trigger a second look.

IF IT DOESN'T

If it doesn't, put your engineering time into detection and response, and let your analysts build their own workflows on a platform that already handles the rest.

See Agentic SecOps in action

Book a working session: bring your stack, we'll show coverage in your environment.

Contact Us →

sales@anvilogic.com | anvilogic.com

ANVILOGIC™ THE AGENTIC SECOPS PLATFORM

Sources: 1. Gartner, "Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027," press release, June 25, 2025.