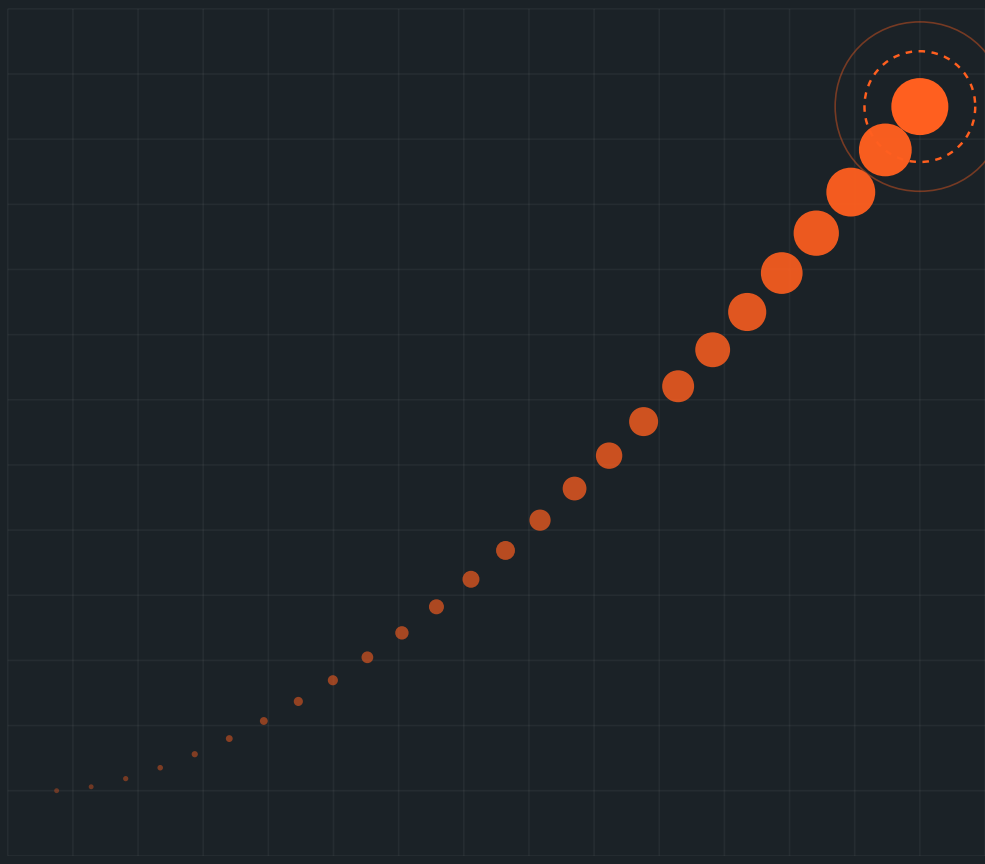
SCALYTICS

WHITE PAPER, OCTOBER 2026



Agents on the Kafka you already run

From sensor to superintelligence: how Lascaris lets AI agents act on live data, with every action checked against your policy and every decision on the record

Alexander Alten, Co-Founder and CTO, Scalytics

Contents

	Summary	0
	Bottom line	0
01	Agents that act can also act wrong	0
02	Why Kafka is the right foundation	0
03	How Lascaris works	0
04	Architecture	0
05	Deployment options	0
06	An open alternative to closed platforms like Palantir	0
07	Use cases	0
08	What the record means for AI Act, DORA and NIS2	0
09	What Lascaris is not	0
10	How to start	0
	About Scalytics	0
	Sources	0

WHITE PAPER

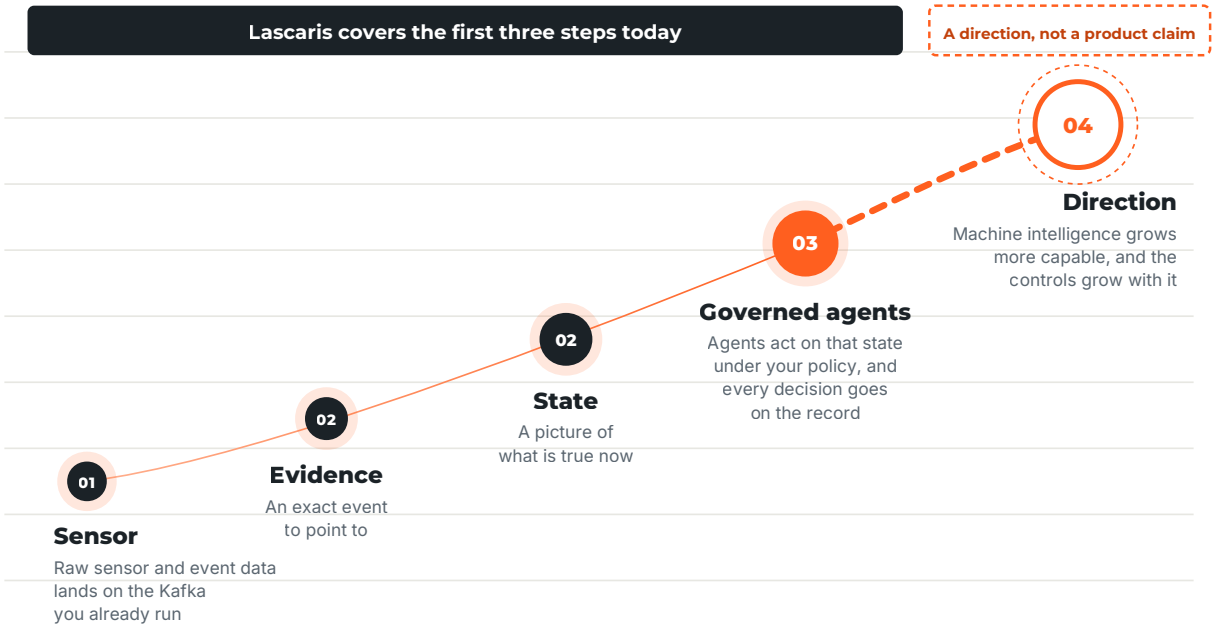
Summary

AI agents are moving from answering questions to taking actions. They hold a payment, change a load profile, reroute a shipment, slow down a robot. The moment an agent can change something real, three questions land on the desk of every platform and data leader. Who stops a bad action? Can you show why the agent acted? Where does your data go?

This paper argues that the answer belongs on the Kafka you already run. Kafka carries your live events. It keeps them in order. It already sits inside your security boundary. What it does not do on its own is decide what an agent is allowed to do with an event.

We think of this as a path from sensor to superintelligence, in four steps. First, raw sensor and event data lands on the Kafka you already run. Second, that data becomes evidence and state: an exact event to point to, and a picture of what is true now. Third, agents act on that state under your policy, and every decision goes on the record. Fourth, over the long run, machine intelligence grows more capable, and the controls grow with it. Lascaris covers the first three steps today. The fourth is a direction, not a product claim, and it only works if the first three are solid.

FROM SENSOR TO SUPERINTELLIGENCE



Lascaris adds that missing part. Every agent action runs through one loop: Sense, Decide, Check, Record. Agents read live events from your topics. They reason with the model you choose. Before anything runs, the action is checked against the policy you set. If it fails the rule, it does not run. Every decision is written to an immutable audit log. The whole loop stays inside your network.

Lascaris is an open, sovereign alternative to closed platforms like Palantir. It runs on the Kafka you already run, inside your network, on Apache 2.0 foundations you can read and run yourself. You can start with it today. This paper describes what Lascaris is designed to be and where it is going.

The paper is for enterprise data and platform leaders and architects who run Kafka today. It covers the problem, why Kafka is the right foundation, how Lascaris works, the architecture, the deployment options, how Lascaris compares with closed platforms, real use cases, and how to put a first agent on live data without betting the business on it.

BOTTOM LINE

> **An agent that acts needs three things a chatbot never needed: fresh state, a hard check before the action, and a record you can show to an auditor. Kafka gives you the fresh state. Lascaris gives you the check and the record, on the same Kafka, inside your network.**

The practical takeaway is simple. Do not start with the model. Start with one event stream, one action, and one written rule that action has to pass. Then look at the audit record and ask whether your risk team would sign it off.

That is the first step from sensor to superintelligence. Machines will take on more decisions over time. The teams that get there safely will be the ones whose agents already run on real events, under written rules, with every decision on the record.

01

Agents that act can also act wrong

- From answering to acting
- A model can be wrong with full confidence
- Four questions to answer before the first agent goes live
- Chat logs are not an audit trail
- Connection is not control

From answering to acting

Most enterprise AI today answers. It summarises a document, drafts an email, finds a contract clause. If the answer is wrong, a person reads it and moves on. The damage is small.

An agent that acts is different. It does not hand a suggestion to a person. It changes a system. It blocks a card, opens a valve, sends a command to a robot, files a report. If it is wrong, the mistake reaches production before anyone reads it.

The value is real. Fraud, outages and supply problems do not wait for a nightly batch. An agent that reacts to the event itself, when it happens, can do work that a report cannot. That is why so many teams want agents on live data. It is also why so many agent pilots stop at the security review.

A model can be wrong with full confidence

Language models are good at sounding right. They are not built to know which internal rule applies to this customer, this asset or this shift. A prompt that says “never move more than 10 percent of load” is a hope, not a control. Context windows get cut. Instructions conflict. A model can write a very reasonable explanation for an action it should never have taken.

So the control cannot live in the prompt. It has to sit outside the model, at the point where a proposed action turns into a real effect.

Four questions to answer before the first agent goes live

When I talk to platform and AI leaders about agents, the same four questions come up. I suggest you have an answer to each one before the first agent touches production.

Who stops a bad action?

Without a hard policy check before the action, the first mistake reaches production.

Can you show why it acted?

Regulators and your own risk team will ask what the agent saw, what it decided and which rule allowed it.

Where does your data go?

Many agent tools send prompts, context and reasoning to an outside service. For regulated data that is often a non-starter.

Can you leave?

Agents built into one closed stack are hard to move. Your models, your rules and your history should stay yours.

Chat logs are not an audit trail

A chat transcript shows what text went into a model and what text came back. That is useful for debugging. It is not a decision record.

A transcript usually cannot tell you which exact event the agent used. It cannot tell you which version of a rule was in force at that moment. It cannot tell you whether a downstream call already went through before a worker crashed and a retry started. And it cannot tell a reviewer whether an approval was still valid after the rule changed.

The same is true for a tool call with a confirmation button. A confirmation answers “may this call run now?” It does not answer “was the evidence still fresh after a two hour wait?” or “did this already happen once?” Those questions need a record that lives longer than the chat session and the worker that ran it.

Connection is not control

Connecting an agent to Kafka gives it access to evidence. Connecting it to tools, for example through MCP, gives it a clear and limited way to call systems. Both are good and both are needed. Neither one decides whether a specific action is allowed, under which rule, right now. That decision, and the record of it, is the part most agent stacks leave open.

02

Why Kafka is the right foundation

- Agents need what is true now
- The log is the evidence
- One spine, no new copy
- Decisions can be events too
- What Kafka does not do

Agents need what is true now

A copilot can live with yesterday's data. An agent that acts cannot. If it reroutes a shipment, blocks a payment or changes a load profile, it has to see the event when it happens, not after the next batch job.

Kafka already carries your orders, payments, telemetry and alerts as they happen. Agents that read the stream see the same live picture your systems do. They decide on fresh state, not on a snapshot that describes a world you already left.

The log is the evidence

Kafka keeps records in topics and ordered partitions, and every record has an offset. That sounds like plumbing. For agents it is the most important property of all.

Because every event has a precise address, a decision can point to the exact record it used: this topic, this partition, this offset. A reviewer does not have to trust a summary of "live data." They can look at the event itself, check what came before and after, and replay the sequence. Freshness stops being a feeling and becomes something you can test.

One spine, no new copy

Most AI patterns start by copying data out of the systems that produce it into a separate platform. That copy has to be secured, governed and paid for. It is also somewhere you no longer fully control.

Agents that read from the platform you already trust and govern need no extra pipeline to a third-party AI service and no second copy to secure. Your Kafka already has owners, access controls, retention rules and monitoring. Agents inherit that work instead of starting a parallel world.

Decisions can be events too

If an agent decision is written back as a new event, it gets the same properties as everything else on the log: ordered, durable, replayable. The audit trail stops being a report you assemble after the fact. It becomes a property of the system. Audit turns from an investigation into a query.

Audit turns from an investigation into a query.

What Kafka does not do

Kafka is a very good source of evidence. It does not decide what that evidence permits. Kafka access controls govern who may read or write a topic. They do not decide whether a fraud agent may hold this payment for this customer under the rule that applies today. That is a business decision, and it needs its own enforcement point and its own record. This is the gap Lascaris fills.

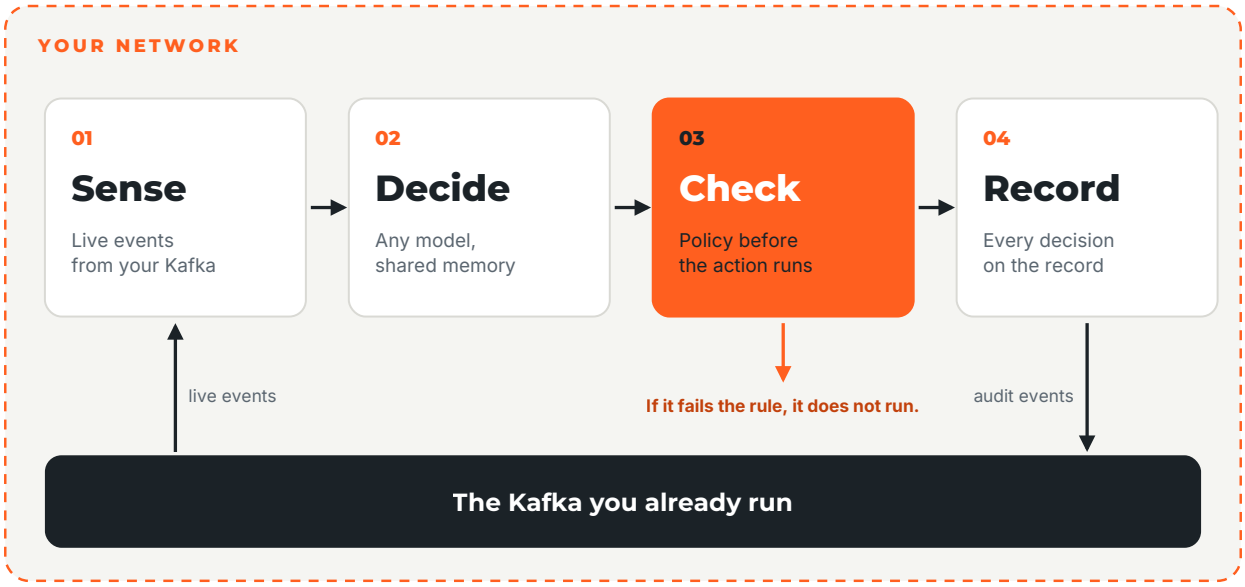
03

How Lascaris works

- Sense: live events from your Kafka
- Decide: any model, shared memory
- Check: policy before the action runs
- Record: every decision on the record
- Inside your network
- One loop, step by step

Lascaris is the foundation your AI agents run on. It installs onto the Kafka you already have and uses it as the spine. If you have no cluster, the Kafka-native engine is included.

Every agent action goes through the same four steps, inside one boundary: your network.



The Lascaris decision loop: Sense, Decide, Check, Record, inside your network.

01 Sense: live events from your Kafka

Agents read live data straight from your Kafka topics. These are the same streams your operations already run on: orders, payments, sensor readings, alerts, robot telemetry. There is no export job and no second copy.

02 Decide: any model, shared memory

Agents reason on live state with any model that has an API, including models you host yourself. If you want the reasoning to stay inside your network, you run the model inside your network too, and Lascaris supports that.

Agents also share memory. A live, shared memory gives every agent the same picture, so a team of agents does not act on five different versions of the truth. Through MCP support, agents can use the tools you already expose.

03 Check: policy before the action runs

Before anything happens, the proposed action is checked against the policy you set. If it fails the rule, it does not run.

This is the core of the design. The model proposes. Your policy decides. The rules are yours, not the model's. They are written down, versioned and enforced at run time, outside the prompt. A model that is confident and wrong still cannot push an action past a rule it fails.

The model proposes. Your policy decides.

HOW POLICIES WORK TODAY.

Each part of Lascaris enforces its own rules. The agent runtime checks every tool call against a risk tier (read, write, high risk) and a list of allowed senders, and its guards can warn on, block or redact personal data and secrets in prompts and outputs. Detection rules are YAML files with SQL, grouped in versioned packs, and the version is stamped on every alert. Rules live in files under Git, so you change a rule by changing the file, you see every change in the history, and you roll back by deploying the previous version. A central policy engine, a policy change workflow and roles for policy authors are on the roadmap, with no release set.

HUMAN APPROVAL.

The runtime has an approval gate in code: a call above the auto-approve tier is posted to the chat channel, and the agent waits for approve or deny from chat, the web UI or a REST call. Each request is stored with tool, tier, arguments, sender, trace id, status and response time. It is switched off in the default settings, and it does not yet record who approved. A full human approval step before action, with the approver on the record, is on the roadmap, with no release set. Until then, the safe pattern for high-risk actions is to let agents recommend and log while a person acts.

04 Record: every decision on the record

Every agent decision is written to an immutable audit log on S3 through kafSCALE, the engine Lascaris runs on, built for the AI Act, DORA and NIS2. The same decision is published to Kafka as an audit event. You can show what the agent tried to do, whether it was allowed and the reason. A denied action is recorded too, so a missing action is never a mystery.

WHAT A RECORD HOLDS TODAY.

Each allow or deny decision is stored with an id, trace id, task id, tool, risk tier, sender, channel, the result, the reason and a timestamp, and the same decision goes to Kafka as an audit event. Kafka topics are append-only, and you set retention as for any other topic. Every decision is also written to S3 through kafSCALE, the engine Lascaris runs on, where it is kept as an immutable audit log. On top of that, compliance reports can be written to object storage with Object Lock in compliance mode, so they cannot be changed or deleted during the retention period.

Inside your network

The whole loop stays inside your network. Lascaris is self-run by your team, on-premise or in your own cloud account, with Scalytics support. A managed option is on the roadmap. Your data and your agents' reasoning stay in your environment. With models hosted inside the same boundary, the data leaving your network is zero bytes.

Zero bytes

With models hosted inside the same boundary, the data leaving your network is zero bytes.

One loop, step by step

The Lascaris product page shows one pass of the loop on a robot fleet. A mobile robot streams its laser scan to Kafka. An agent sees an obstacle in a zone and works out another path. The policy for that zone caps speed at 0.5 meters per second. The agent's proposed command of 0.4 meters per second passes the rule, so it is published to the robot. The decision and the policy result are written to the record.

Had the agent proposed 0.7 meters per second, the check would have failed and the command would never have reached the robot. The record would show the proposal, the rule and the denial.

The same shape works for a payment, an insurance claim or a grid asset. The event changes. The model may change. The loop does not.

04

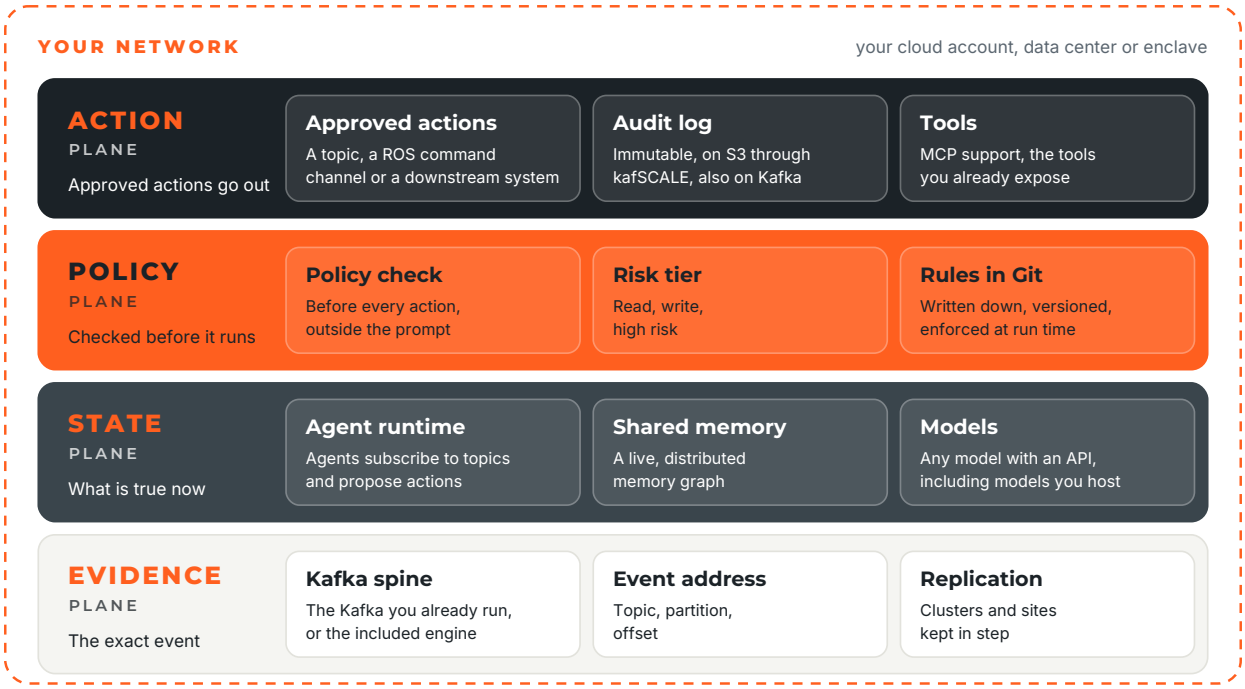
Architecture

- The parts
- How the pieces fit
- Design principles
- Trust in what agents may do

The parts

Lascaris is one distribution with a small number of parts, shipped together, version-locked and tested as one.

PART	WHAT IT DOES
Kafka spine	The included Kafka-native engine, S3-native and wire-compatible, certified today. Confluent Platform 7.6 up to 8.0 and Redpanda 23 and later are expected to work, not yet certified.
Agent runtime	Coordinates the agents, enforces policy before every action, and records every decision.
Shared memory	A live, distributed memory graph every agent reads from and writes to.
Replication	Keeps clusters and sites in step, for resilience and reach across locations.
Audit log	An immutable record of every allow or deny decision, written to S3 through kafSCALE and also published to Kafka. Compliance reports can also sit on Object Lock storage you control.
Models	Any model with an API, including open models you host next to your data.
Tools	MCP support, so agents can use the tools you already expose.



Lascaris reference architecture inside your network.

How the pieces fit

Events flow from your producers into Kafka as they do today. Lascaris agents subscribe to the topics they need. When an agent wants to act, it does not call a system directly. It proposes an action. The runtime checks the proposal against the active policy. Only an approved action is sent on, for example to a topic, a ROS command channel or a downstream system. The decision, approved or denied, is written to the audit log.

Nothing in this path needs to leave your network. The engine, the runtime, the memory, the audit log and the models can all run in the same account, data center or enclave.

Design principles

A few principles shape the architecture. They come from running data platforms where every decision had to be accounted for.

01

Keep observation separate from authority.

An agent that can read a topic should not, by that fact alone, be able to change the systems behind it. Reading is evidence. Acting needs permission.

02

Ask for a capability, do not hand out a tool.

A planning agent that needs to know if a spend fits the budget does not need the ledger. It needs a bounded answer from someone allowed to give it. Requests and answers travel over Kafka. Authority stays with the part of the system that owns it.

03

Make every effect safe to retry.

Workers crash. Networks drop. A correct decision can still cause a duplicate effect if a retry runs twice. So each action needs a stable key and a recorded status, so a retry can see whether the effect already happened.

04

Treat the decision as the thing worth keeping.

A pile of messages is not a decision record. The record has to tie together the evidence, the rule, the result and the effect, so a reviewer can follow the chain without asking a model to remember.

A good decision record answers four questions:

RECORD ELEMENT	QUESTION IT ANSWERS	FAILURE IT PREVENTS
Source event address (topic, partition, offset)	What evidence was used?	Context you cannot verify
Policy and version, with the check result	Why was it allowed at that moment?	Acting on an outdated rule
Action key and effect status	Did the effect happen once?	Duplicate actions after a retry
Review state and replay pointer	May it resume now?	Blind recovery after a failure

Out of the box, Lascaris writes the policy result with tool, risk tier, sender, trace id and reason. Source event address, policy version in the record, action key, effect status and replay pointer are built per use case during onboarding today.

Trust in what agents may do

Policy decides whether a single action may run. A second question is what an agent is allowed to do at all. Scalytics describes a capability layer in which every skill an agent can use is packaged, signed and given a verifiable identity, so its origin and permitted use can be checked before it runs, and it can be revoked on demand.

Signed, revocable skills exist today in the Scalytics skill tooling: signatures checked offline against pinned trust roots, signed revocation lists and a fleet kill switch that fails closed. Integration into the agent runtime is on the roadmap, with no release set. In the runtime today, installed skills are pinned by hash and can be switched off in config.

05

Deployment options

- Where it runs
- Your Kafka or ours
- Open core, supported distribution

Where it runs

You choose where Lascaris runs. In every option the data, the reasoning and the audit log stay inside your boundary.

OPTION	WHO RUNS IT	WHERE	TYPICAL FIT
Managed (roadmap)	Scalytics manages it	Inside your own cloud account	Teams that want agents in production without growing the platform team
Self-run on-premise	Your team, with Scalytics support	Your data center, your hardware	Teams with existing on-premise Kafka and strict data rules
Air-gapped	Your team, with Scalytics support	A disconnected enclave, no outbound connection required	Defense, critical infrastructure and other high-security sites

SELF-RUN TODAY.

Lascaris is tested on a single k3s node with Kubernetes 1.36 and on KIND with Kubernetes 1.35, with S3-compatible storage. A development box needs at least 4 CPUs, 8 GiB RAM and 30 GiB disk. The bundled user apps (NiFi, Grafana, JupyterLab) request about 3 CPUs and 11 GiB. Multi-node, high availability, OpenShift and the big managed Kubernetes services are not tested yet. Releases so far are release candidates for pilots, and the first signed stable release is next. Version 1.0 has no built-in authentication, so run it inside a protected network; identity through Keycloak is planned for 1.1.

MANAGED (ROADMAP).

The managed option is planned for IBM Cloud, Google Cloud and Cloud&Heat. For customers who need an EU provider outside the US CLOUD Act, the candidates are STACKIT, Open Telekom Cloud and IONOS. No managed deployment runs today.

Your Kafka or ours

On the Kafka you already run. Already run Confluent or Redpanda? Lascaris is built to run on it and use it as the spine. Confluent Platform 7.6 up to 8.0 and Redpanda 23 and later are expected to work today, and we test your setup with you in the pilot. You keep your cluster, your operations team and your vendor relationships. Scalytics is a Confluent partner, and Lascaris is meant to sit on top of your setup, not replace it.

Engine included. No cluster yet, or a site that needs its own? The Kafka-native engine ships inside. It is S3-native and wire-compatible, so you stand up the whole platform as one install with no cluster to build first.

> **KAFKA SUPPORT TODAY.**
>
> The included engine is the certified backend. Confluent Platform 7.6 up to 8.0 and Redpanda 23 and later are expected to work and are not yet certified. Amazon MSK is not tested yet, because IAM authentication is not in the charts. Against the included engine, producers must set enable.idempotence=false, since it has no transactions, and some clients read by partition assignment instead of consumer groups. Tested clients: Apache Kafka 3.9 CLI, Confluent 7.6.1 and the Java client 3.9.2. SASL and ACLs on the included engine are planned for its 2.0 release. To link external clusters, replication supports SASL PLAIN, SCRAM and GSSAPI.

Open core, supported distribution

The components of Lascaris are open source under Apache 2.0. You can read the code and run the parts yourself. Lascaris is the distribution: the same parts, integrated, tested, hardened and maintained as one. Think of a Linux distribution, applied to the agent stack.

	OPEN-SOURCE COMPONENTS	LASCARIS
Integration	You assemble the components	**Shipped integrated, ready to run**
Versions	You manage cross-release compatibility	**Version-locked and tested together**
Security	You harden it yourself	**Security-maintained for you**
Support	Community and self-support	**Defined SLAs, one contract**
Maintenance	You patch and backport	**Long-term support and backports**
Kafka engine	Runs on your Kafka	**Engine included**
License	Apache 2.0, free	**Commercial subscription**

This is also the answer to “can we leave?” Your models, your rules and your decision history stay in your environment, on open foundations. Nothing is locked inside a platform you cannot run yourself.

For the subscription model and pricing, talk to us at **scalytics.io/lascaris**.

06

An open alternative to closed platforms like Palantir

- Where the two differ
- What design partners say

Our design partners in defense compare Lascaris with Palantir Gotham. So it is worth being direct about how the two differ.

Palantir describes Gotham as "the operating system for defense decision making." Its annual report says Gotham has surfaced insights for global defense agencies, the intelligence community and disaster relief organizations for over a decade. Palantir also sells Foundry for enterprises, Apollo for software delivery, and AIP, which brings AI models into Gotham and Foundry. These are serious products with a long track record.

Lascaris is built for the same kind of operator: people who run high-stakes operations on live data and have to keep control of the outcome. It takes a different path.

Where the two differ

- **Open core instead of proprietary software.** Palantir's terms for its Google Cloud offering define its software as "Palantir proprietary software" and give customers access during the order term, solely for their internal business purposes. Palantir's AIP terms forbid customers to reverse engineer or attempt to discover its source code. The Lascaris components are Apache 2.0. You can read every line, audit it, and run it yourself. The open components keep working without a contract with us.
- **Your Kafka as the spine, not a new central platform.** Palantir's annual report says Gotham and Foundry "transform massive amounts of information into an integrated data asset." Lascaris does not ask you to build that new asset first. It runs on the Kafka you already run. Your events stay in the streams your teams already operate, and every decision is written back to infrastructure you own.
- **Inside your network, all of it.** Palantir also runs in customer environments. Its AIP page says "Run LLMs and other models on your private network with full control." So location alone is not the difference. Lascaris is built so the whole stack runs in your environment, including air-gapped sites: the Kafka engine, the agent runtime, the policy check, the audit log and the models. It runs on code you can inspect, built by a European company.
- **No lock-in.** Your rules, your models and your decision history stay on your Kafka and your storage. If you stop working with Scalytics, the open components, your topics and your audit log are still yours.

What design partners say

We do not claim feature parity. Design partners who know Gotham tell us Lascaris does not yet have all of its heavier legacy features, and we would rather you hear that from us. Here is their feedback, exactly as published on the Lascaris page.

> "We have been a design partner testing Lascaris as a command system for drone and intelligence operations. Compared to Palantir Gotham it is still missing some of the heavier legacy features. That part is real. But the core ontology modeling, real time telemetry fusion, and analyst workspace already feel solid and highly usable.
>
> What stands out is how responsive the team is. They actually listen and move fast to fix things. Implementation and support have been much smoother than what we experienced with bigger vendors. Running it fully sovereign in our own environment is a big plus.
>
> The platform is evolving fast, and they are clearly determined to build something better for users who want absolute control."
>
> — **DEFENSE DESIGN PARTNER**

> "Design partner comment on Lascaris from Scalytics.
>
> It lacks some of the legacy modules compared to Gotham, but the basics work flawlessly (ontology, provenance, analyst views). The team is fast and helpful which is refreshing. We set it up completely in our own environment without issues. A much more agile foundation. We use it as command control for intelligence operations in addition to legacy systems. The autonomous agents are the absolute winner here."
>
> — **SYSTEMS ENGINEER**

> "We joined as a design partner to test Lascaris in a real operational setting. It does not carry the heavy feature bloat of Palantir, but I think that is exactly the point, since they have a decentralized approach.
>
> On the positive side the platform already delivers excellent fusion capabilities at the edge, and the Scalytics team has been exceptional with implementation and ongoing support. They are hungry to improve it and it shows.
>
> The ability to keep everything sovereign and deployed exactly where we need it is a real advantage for us."
>
> — **OPERATIONAL ARCHITECT**

Source: **scalytics.io/lascaris**, "Early operator feedback."

07

Use cases

- Financial services and insurance
- Telecommunications
- Health
- Energy and critical infrastructure
- Defense and national security
- Robotics and fleet operations
- Space

The use cases below come from the domains Scalytics serves today. Each one follows the same loop. The event changes. The rule changes. The record is always there.

Financial services and insurance

Real-time fraud. Fraud agents watch the transaction stream and flag or hold suspicious payments as they happen. The check makes sure a hold stays inside the limits your risk team set. Every decision is recorded for the regulator.

Claims triage. Agents screen insurance claims on the stream. They fast-track clean claims and flag suspicious ones for a person. Every decision is on the record, so a customer complaint can be answered with facts.

Continuous compliance. Agents monitor operational streams against policy and catch a breach as it happens, instead of in the next quarterly review. The decision record gives you evidence for DORA, NIS2 and AI Act questions.

Telecommunications

Network fraud. Agents detect SIM-swap, subscription and revenue fraud on live network and call detail record streams. Every decision is recorded under your data-retention rules.

Health

Clinical alarm triage. Agents separate real clinical alarms from noise on the live monitor stream. Every suppression is on the record, so clinical governance can review exactly what was held back and why.

Patient monitoring. Agents watch live vitals where patient data already lives and surface the patients who need attention, inside the hospital network.

Energy and critical infrastructure

Demand and peak management. Helios, the energy management solution built on Lascaris, forecasts demand, price and generation from your own SCADA and telemetry. It then dispatches storage, generation and flexible loads to cap the peak that sets your bill. Every decision is a record you can show to a grid operator or a regulator.

Grid and plant anomaly detection. Agents run on high-volume telemetry from plants, substations and field assets. Every alert is traceable back to the source signal, and OT systems are not exposed to the open internet.

Recommend first. In grid operations, agents recommend and log, and the operator decides and moves the asset. A built-in approval step that records the approver is on the roadmap.

Defense and national security

Sensor fusion and logistics. Agents ingest radar, optical and tracking data inside an air-gapped enclave. They correlate threats in shared memory and pass every recommendation through policy before it is committed to the mission log. Every decision is recorded for after-action review.

Robotics and fleet operations

Fleet safety. Agents ingest ROS telemetry from many robots in real time, detect obstructions and correct speed envelopes through governed command topics. Hard safety policy controls every command, as in the example in section 3.

Space

Satellite telemetry. Agents handle bursty downlink telemetry across ground station windows, track orbital drift against live debris catalogs and calculate maneuver adjustments that must pass hard propellant budget limits before they are accepted.

Design partner feedback in their own words is in section 6.

08

What the record means for AI Act, DORA and NIS2

European rules now expect organisations to explain and evidence automated decisions, not only produce them. DORA has applied to financial entities and their critical ICT providers since January 2025. NIS2 extends cybersecurity and incident reporting duties across critical sectors. The EU AI Act adds record-keeping, logging and human oversight duties for high-risk systems, phasing in through 2026 and 2027.

The common need behind all three is the same: show what happened, show why, and show who was in control. The Lascaris loop is built around that need.

Sense	ties every decision to an exact event, so evidence is precise and replayable.
Check	proves a written rule was applied before the action, not explained after it.
Record	writes every decision, approved or denied, to an immutable audit log on S3 through kafSCALE, and publishes it to Kafka. Compliance reports can also be locked on object storage.
Inside your network	keeps data, reasoning and evidence in your jurisdiction.

The record does not make a bad rule good, and it is not legal advice. What it does is make every automated decision inspectable, which is the condition for defending it.

We map the record to specific AI Act, DORA and NIS2 duties together with each customer's compliance team. We do not claim a certification here.

09 What Lascaris is not

Being clear about limits saves everyone time.

- **Not a chatbot or a SaaS copilot.** Lascaris is for operators who run agents that act on live data and have to keep control of the outcome.
- **Not a Kafka replacement.** It runs on the Kafka you have. The engine is there for when you have none.
- **Not a fix for bad policy.** If a rule is wrong, Lascaris will enforce the wrong rule and record it faithfully. The record makes the mistake visible. It does not make it right.
- **Not free of cost.** A policy check and a decision record add steps compared with a direct tool call. For a local, read-only calculation that touches nothing sensitive, a direct call can be the better choice.

---

i We have not published a latency figure for the check and the record. Measure it on your own workload during the pilot. The guardrail detectors run as separate Kafka consumers outside the request path by design.

10 How to start

Do not start with a platform program. Start with one agent, one stream and one rule.

01 Pick one use case on your Kafka.
Choose an event stream that matters and an action an agent could take on it. A fraud hold, an alarm suppression, a load shift.

02 Write the policy it has to pass.
Put the limits in writing: amounts, zones, speeds, approval thresholds. Writing the rule down is often where the real questions show up.

03 Run the loop and read the record.
Let the agent sense, decide, get checked and get recorded. Then put the audit record in front of your risk or compliance team and ask whether they would sign it off.

That is exactly what we do on a first call. We pick one use case on your Kafka, set the policy it has to pass and show what the audit record looks like.

A first pilot runs on a single Kubernetes node with the included engine: one agent, one stream, one rule.

Put your first agent on live data, safely.

Start at **scalytics.io/ai-agents-on-kafka**.

Or start with the AI Architecture Score on the same page to see where your stack stands today.

About Scalytics

Scalytics builds Lascaris, the sovereign AI agent platform for operators that run autonomous agents on live data. Scalytics was founded by the original inventors of Apache Wayang, now an Apache top-level project. The team has held senior and executive roles at E.ON, Allianz, Cloudera, Scout24 and McKinsey, and includes veterans of the German armed forces. Scalytics is a registered Confluent partner.

About the author. Alexander Alten is Co-Founder and CTO of Scalytics, and co-creator and PMC member of Apache Wayang. Before Scalytics he worked on data platforms at Allianz, Cloudera and E.ON. He writes about high-throughput systems and AI agent infrastructure at **novatechflow.com**.

Sources

- Scalytics, Lascaris product page: https://www.scalytics.io/lascaris
- Scalytics, AI agents on Kafka: https://www.scalytics.io/ai-agents-on-kafka
- Scalytics, What is a Sovereign Decision Fabric?: https://www.scalytics.io/sovereign-decision-fabric
- Scalytics, Regulated industries: https://www.scalytics.io/solutions/regulated-industries
- Scalytics, Energy and Helios: https://www.scalytics.io/solutions/energy
- Scalytics blog, Why Kafka and MCP Are Not Enough for Autonomous AI Agents (September 2026): https://www.scalytics.io/blog/why-kafka-mcp-not-enough-autonomous-ai-agents
- Scalytics blog, Data Sovereignty Fails the Moment Context Becomes Action (October 2026): https://www.scalytics.io/blog/data-sovereignty-context-to-action
- Scalytics blog, Stop Giving AI Agents Direct Access to Your Tools (October 2026): https://www.scalytics.io/blog/agent-capability-authorization-kafka
- Lascaris decision loop explainer video script: scalytics/sales, videos/explainers/lascaris-decision-loop/script.md
- Palantir, Gotham platform page: https://www.palantir.com/platforms/gotham/
- Palantir, AIP platform page: https://www.palantir.com/platforms/aip/
- Palantir Technologies, Form 10-K for fiscal year 2024: https://www.sec.gov/Archives/edgar/data/1321655/000132165525000022/pltr-20241231.htm
- Palantir, Terms and Conditions for GCP: https://www.palantir.com/assets/xrfr7uokpv1b/2ajxcAAipDS2EAkFxZZvDa/ed556b05d8ef5d80b91a0e7bf8aff559/Palantir_Terms_and_Conditions_for_GCP.pdf
- Palantir, AIP terms (customer use of AIP): https://palantir.pactsafe.io/versions/649b3192ed79b18a284396fe.pdf

LASCARIS

Put your first agent on live data, safely.

Start at **scalytics.io/ai-agents-on-kafka**

SCALYTICS

scalytics.io