OSA CON 26

What Breaks When Open Source Data Platforms Become Agent Infrastructure?



Siddharth Arun

Senior Member of Technical Staff

Salesforce

Nov 2, 2026 | AWS Builder Loft & Online (osacon.io) | Free

Same data. A different responsibility.

What changes when the consumer of a data platform can act?



A green pipeline means the job finished.

It does not mean the agent chose a safe action.

Who needs to care?

When data turns into action, guarantees cross team boundaries.



01

Data engineers

Freshness, completeness, provenance



02

Platform & SRE

Safe execution and recoverability



03

AI engineers

Tool selection, runtime plans



04

Security & governance

Authorization and blast radius

The new variable: who chooses the next action?

Traditional automation has defined paths; agents can select paths at runtime.

TRADITIONAL AUTOMATION

IF alert → restart → Check health

Known branches, encoded rules

Still needs retries and idempotency

AGENTIC EXECUTION

Agent reasons → Chooses tool

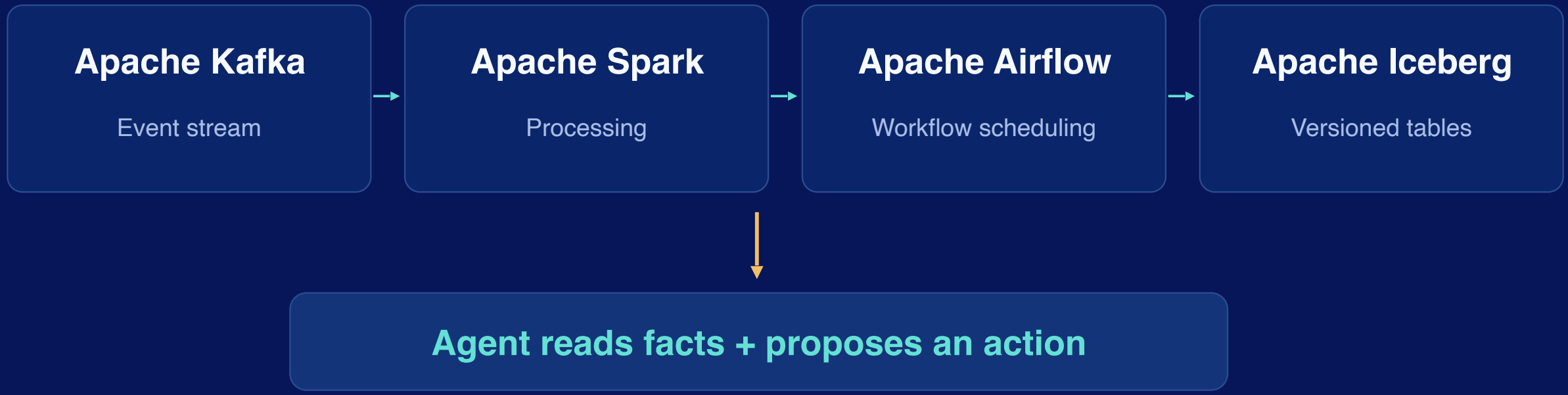
Runtime-chosen actions + parameters

Needs independent policy checks

Same distributed-systems foundations. New autonomy boundary.

The open-source data backbone is already here

These tools supply facts and execution primitives—not agent permission by themselves.



Missing boundary: validate context, permissions, and execution

Four industries. One shift in responsibility.

In each case, the agent chooses an action that affects the real world.

01

Cloud migration

Chooses a target region while capacity changes

02

Incident response

Chooses a chain of production remediation steps

03

Advertising

Changes spend before performance data matures

04

Customer refunds

Interprets policy and chooses a payout

A valid plan can become an invalid action

The agent selects the destination. Infrastructure must verify it at commit time.

10:00 Agent reads capacity → **10:02** Region B fills up → **10:03** Agent requests move



Action boundary checks current capacity + placement rules

Then reserve / compare-and-set before executing

AGENTIC RISK: a runtime-selected plan can become stale

One bad diagnosis. Multiple autonomous changes.

The critical risk is not duplicate retries—it is an unchecked chain of decisions.

Alert
Latency rises

→

Agent: restart
No improvement

→

Agent: scale
Cost increases

→

Agent: config
Wider impact

Stop-and-check after every tool call

Allowed actions · parameter limits · blast radius · verification · approval

The data is late. The budget change is immediate.

An illustrative conversion-attribution timeline.

10:00 AM · INCOMPLETE VIEW

2 purchases

12:00 PM · LATER VIEW

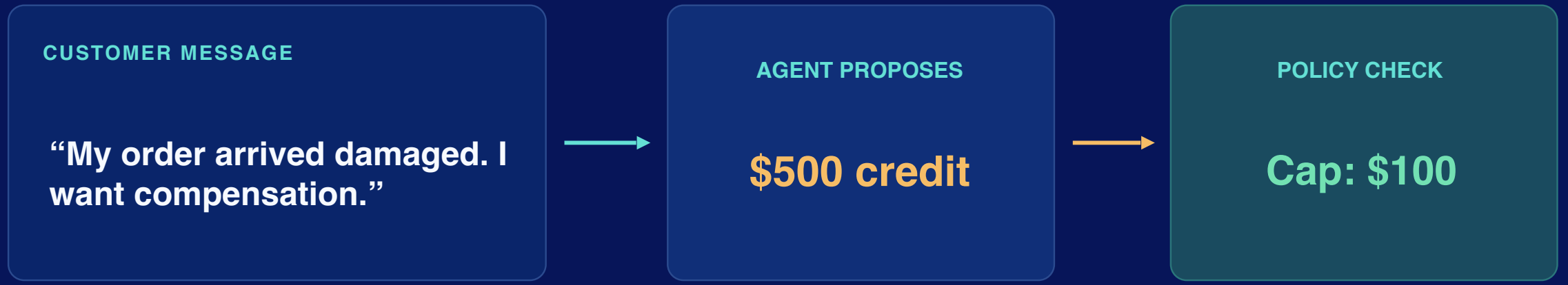
22 purchases

Agent cuts spend at 10:00 — before 20 delayed conversions are visible

SAFE ACTION: enforce event-time maturity, confidence and budget limits

Natural language does not grant financial authority

The agent interprets policy, chooses a tool, and selects an amount.



Policy controls the amount. Idempotency prevents duplicate payments.

These are familiar primitives—combined under autonomy

The novelty is how runtime decisions connect the failure modes.

01 Changing data

Plan uses an expiring view

02 Dynamic execution

Tool sequence chosen at runtime

03 Side effects

Actions need authority + safety

04 Decision provenance

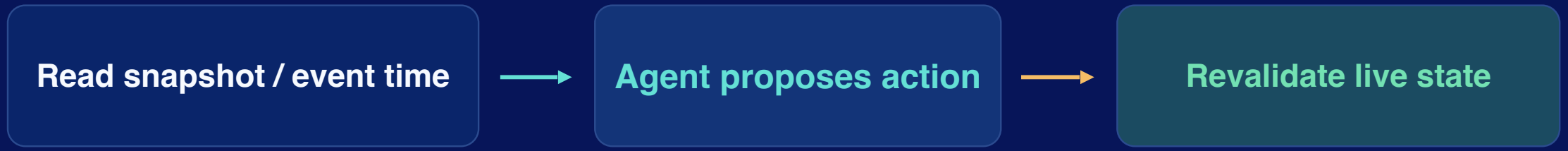
Why did the agent choose this?

05 Selective recovery

What can safely be resumed?

Context can expire before the action commits

A valid answer at 10:00 can be a dangerous instruction at 10:03.



What to enforce

- Max context age • Version / snapshot ID
- Compare-and-set or capacity reservation at commit time

The agent writes its plan at runtime

The toolbox is finite; the sequence and arguments are not always predefined.

MODEL OUTPUT

1. **Restart service**
2. **Scale replicas**
3. **Change routing**



THE CONTROL PLANE ENFORCES

- Typed tool APIs and parameter bounds
- Least privilege and resource allowlists
- Per-action policy and blast-radius check
- Human approval for high-risk choices

Reliable execution is not the same as permission

Idempotency still matters. It does not tell us whether the decision is allowed.

EXECUTION SAFETY

Did this approved action run once?

Idempotency key · durable result · reconciliation

AGENT AUTHORITY

Should this action be allowed at all?

Policy · identity · scope · limits · approvals

Both gates are required before financial or infrastructure side effects.

Trace the reason, not just the API call

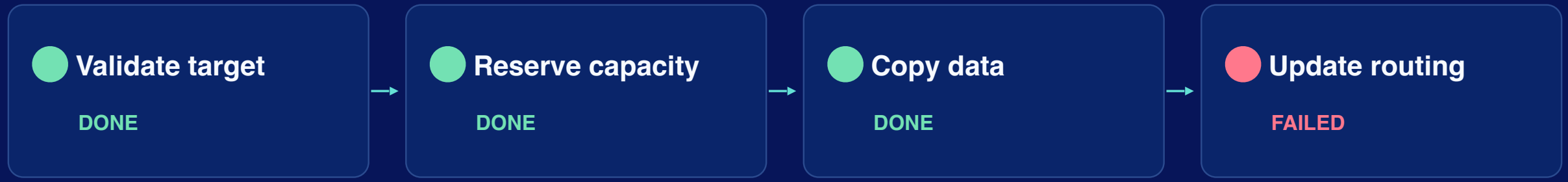
A tool trace tells us what ran. Decision provenance tells us why.



Join decision and action IDs with OpenTelemetry + OpenLineage

Replay the failed step, not the whole world

Persistent state is necessary; side effects still need explicit reconciliation.

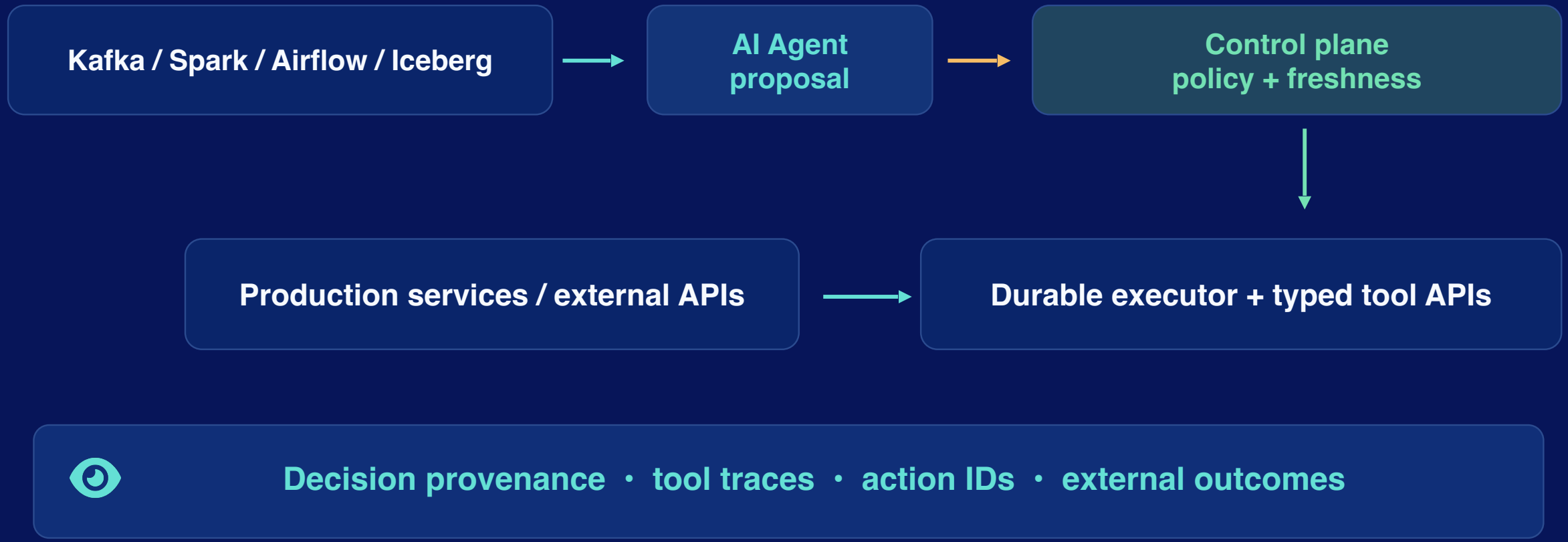


Reconcile external state, then resume only what is safe

Re-plan when facts changed; do not blindly replay old autonomous decisions.

A bounded-autonomy control plane

The agent proposes. Independent infrastructure decides whether and how to commit.



The critical boundary: propose → verify → commit

Make these controls deterministic and independent of the model.



Reject, ask for approval, or require re-planning when the checks fail.

Use open source for the guarantees it actually provides

Useful primitives for data, telemetry, and durability—not action authorization.

Apache Kafka	Event delivery / timestamps
Apache Spark	Compute and event-time processing
Apache Airflow	Predefined DAG orchestration
Apache Iceberg	Table snapshots / versioned data
OpenLineage	Job + dataset provenance
OpenTelemetry	Traces / logs / metrics correlation
Temporal	Durable execution / workflow history

Still required: application policy, action validation, and side-effect reconciliation

The missing glue is where failures hide

Avoid claiming that one open-source component solves end-to-end agent safety.

- ⚠ **Green pipeline ≠ safe decision**
- ⚠ **Snapshot ID ≠ live permission**
- ⚠ **Trace ID ≠ explanation of intent**
- ⚠ **Durable retry ≠ authorized action**

Define an explicit contract between the agent and production.

Increase autonomy only as controls prove themselves

A safe rollout earns write privileges with observed evidence, not model confidence alone.

01

Observe only

Record decisions

→

02

Recommend

Human approves

→

03

Gated actions

Low-risk auto

→

04

Bounded autonomy

Policy + rollback

Track: rejected actions · stale-context blocks · approvals · recoveries

Three engineering principles to take home

The agent chooses. The platform remains accountable.

01 **Treat data as an action dependency**

Freshness, completeness, and provenance are execution preconditions.

02 **Separate reasoning from authority**

The model proposes. Independent controls authorize and commit.

03 **Remember decisions and side effects**

Recover from actual state—not just a restarted prompt.

THANK YOU

What should an agent never change without a safety check?

Siddharth Arun

Senior Member of Technical Staff • Salesforce

linkedin.com/in/siddhartharun7



SCAN TO CONNECT