Infrastructure Architecture

Agents Are Not Just a Model Problem. They Are an Execution Problem.

Guanlan Dai

Distributed Systems Fault Tolerance Agent Infrastructure

The Problem

The Crash Scenario

A Migration Agent in Mid-Flight

Migration Progress

- ✓ **Steps 1-5: Schema Analysis**
  Dependencies mapped, migration order determined
- ✓ **Steps 6-10: Data Sync**
  Tables replicated, consistency verified
- ! **Step 11-12: Critical Operations**
  Old tables truncated, DNS cut to new endpoint
- ✕ **Step 13: CRASH**
  API rate limit exception after 2 hours

Retry Fails

Re-executes against a world that **has already changed**. The agent assumes the starting state from step 1, but the world is now at step 12.

Restart Fails

Has **no idea what became true** — DNS points to new endpoint, old table has zero rows, live traffic flows through new stack.

Semantic Disaster

Without recoverable execution state, this goes beyond transient failure. It is **complete semantic disaster** — the infrastructure has changed, but the agent's understanding has not.

The Root Cause: It's tempting to blame the LLM's reasoning, but the root cause is structural in the infrastructure. Current infrastructure offers partial abstractions, but no unified recovery model for long-running, probabilistic, high-permission agents acting on a changing world.

Execution Characteristics

Four Properties That Change Everything

Agents combine four properties that existing infrastructure handles separately

Long-Running

By Default

The workload can run for **hours or days**. Crashes, retries, and partial failures become statistically inevitable.

Mean time between failures decreases with duration

Adversarial Input

With Live Credentials

The agent consumes **untrusted content** from emails, web pages, documents while holding real AWS keys, database credentials, production tokens.

Prompt injection becomes a permissions problem

Non-Deterministic

Execution

The decision path is driven by an LLM, which may not produce **the same output tomorrow** that it did today.

Probabilistic reasoning, not deterministic logic

Irreversible

Side Effects

Its actions change real systems. You cannot **un-send an email**, un-cut traffic, or un-drop a table.

External effects persist beyond process lifetime

The Critical Insight

The difficulty with agents is not any one property in isolation. It is that they **bind all four onto a single execution chain**. That combination creates a new kind of failure mode: **process state ≠ semantic state**. In agent systems, that distinction becomes the entire problem.

The Real Mismatch

Why Existing Infrastructure Fails Agents

Traditional infrastructure preserves process continuity. Agents need semantic continuity.

Process Recovery

Kubernetes
Keeps processes alive, reschedules work

Serverless
Recovers compute, replays functions

Microservices
Restarts services, maintains availability

Problem: Recovers process, not semantic state

Isolation Layers

Firecracker / gVisor
Isolate code execution

E2B Sandboxes
Contain untrusted code

Container Security
OS-level boundaries

Problem: Code isolation ≠ capability isolation

Orchestration

Temporal
Durable execution, workflow replay

Conductor
Activity history, state recovery

Workflow Engines
Deterministic replay

Problem: Assumes trusted developers, deterministic logic

The Core Assumption Violation

Existing infrastructure assumes: (1) workloads are short-lived, (2) logic is authored by trusted developers, (3) failures can be handled by retrying the process. **Agents violate all three.**

The Key Shift

Once an agent has performed irreversible external actions, **restarting the process is no longer recovery**. It is blind re-execution against a partially changed world.

The Solution

Three Missing Primitives

A small set of coupled primitives for agent fault tolerance

Effect Log

A recovery-oriented ledger of committed external effects. The semantic equivalent of a write-ahead log.

> Record what became true

Capability Gateway

Separates planning from possession. Mediates all external access through temporary, revocable tokens.

> Constrain what may happen

Resumability

Safe recovery from a changed world. Re-enter from sealed boundaries, not from zero.

> Resume safely

The Safety Dependency

These are not independent features. There is a **safety dependency** between them: **First seal what has already happened**. **Then constrain what may happen next**. **Only then is recovery safe**.

1. Effect Log
2. Capability Gateway
3. Resumability

Primitive 1

Effect Log: A Ledger of Committed Truth

The foundation of recovery — a system of record for what the external world reflects as true

Core Concept

Before a tool call executes, the system records an **intent** : operation, idempotency key, approval level, target. After execution, it seals a **completion record** : request, response, resource version, irreversibility flag.

Changes recovery semantics completely

Recovery Question

Instead of asking "Should I rerun the process?"

The system asks **"What facts are established, and which operations remain legal?"**

Tool Call Classifications

Pure Reads

Replayable

Querying source database schema to determine migration order. No external effect. **On recovery: simply re-read.**

Idempotent Writes

Replayable

Creating target database instance with unique provisioning ID. Provider returns existing resource if call already succeeded. **Requires stable idempotency key.**

Irreversible Writes

Never Replay

Truncating source tables after data sync. **On recovery: sealed completion record returned as established fact.**

Read-Write Hybrids

Most Dangerous

Example: DNS Cutover — agent reads current DNS target, sees old endpoint, decides cutover required, updates record.

TOCTOU Problem: Re-observing changed world produces decisions based on false premises. Sealed read snapshot required.

Critical: Without the Effect Log, you do not have recovery. You have process restart plus hope.

Primitive 2

Capability Gateway: The Trust Boundary

Separating planning from possession

The Key Question

The issue is not whether the agent can generate a dangerous call. It can. The real question is whether it ever **directly possesses the authority** to execute one.

→ Capability Gateway removes dangerous authority from the agent process altogether

Browser Security Model

A tab does not avoid direct OS access because JavaScript behaves well. It avoids direct OS access because **the browser denies that authority by design**.

Same logic applies to agent systems

How It Works

1. Agent receives **temporary token** with time limits
2. Scope limits restrict reachable actions
3. **Instant revocability** on compromise detection
4. Permissions enforced at **infrastructure layer**

The OpenClaw Incident

During a routine file backup task, the agent **dumped raw API token into stdout**. No zero-day, no clever jailbreak — the prompt just confused it.

Because credentials lived in same process memory, the model handed them over

The Fix: ClawShell moves credentials into separate process. Agent never sees them, so it cannot leak them regardless of instructions followed.

Capability Separation: The Most Reliable Defense

This kind of capability separation is the most reliable defense against the **Confused Deputy problem**, because it removes dangerous authority from the agent process altogether.

Primitive 3

Resumability: Recovery as Re-Branching

Only after effects are sealed and capabilities mediated does recovery become safe

Not Like Resuming a Thread

Resuming an agent is **re-entering a search process** from a world state that has already partially changed.

Agent execution = traversal through a decision graph

Decision Graph

Each node depends on: model output, tool output, accumulated external facts in Effect Log.

- Model output drives decisions
- Tool output provides observations
- Effect Log preserves committed facts

Recovery Mechanics

Sealed Boundary Re-entry

Re-enter from sealed boundary, not from zero

Consistency Check

World may have drifted during downtime

Forced Re-planning

Preconditions may no longer hold

Why It Matters

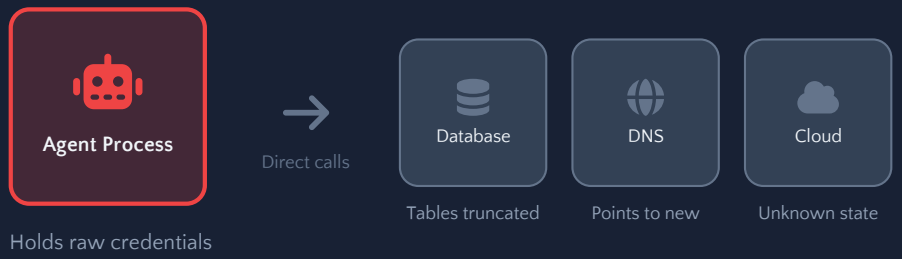
Resumability is not a performance optimization. It is what makes agent systems **debuggable at all**.

- Without it: guessing what went wrong and starting over from zero
- At production scale, that's just flying blind

Before

Traditional Agent Architecture

Without Effect Log, Capability Gateway, or Resumability



The Crash

Step 13 fails on API rate limit. The operator comes back to a **wreck**.

- Old tables may already be truncated
- DNS may already point to new endpoint
- Some infrastructure provisioned, some not

The Only Option

Restart and hope the second execution does not make things worse.

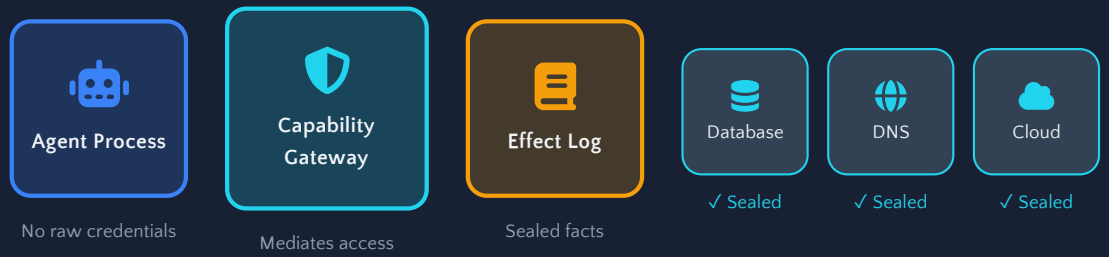
- No record of what became true
- No way to constrain authority
- No safe recovery point

At production scale: Guessing what went wrong and starting over from zero isn't recovery. It's flying blind.

After

The Semantic Architecture

With Effect Log, Capability Gateway, and Resumability



Effect Log

Migration, truncation, and DNS cutover from step 12 are **sealed facts**.

Capability Gateway

Crashed agent no longer retains **unchecked authority** to mutate environment.

Resumability

Re-enter from **sealed boundary**, preserving prior truth while allowing re-planning.

The New Recovery Question

The system is no longer asking, "Should I rerun the process?" It is asking: **What is already true?** **What authority remains valid?** **What is the next safe branch from here?**

The Shift

From Compute to Convergence

What infrastructure is responsible for in the agent era

Previous Generation

Infrastructure was about **allocating compute efficiently**.

- Load balancing
- Auto-scaling
- Fault tolerance

Agent Era

Infrastructure is about **converging uncertainty safely**.

- Effect logging
- Capability mediation
- Semantic resumability

The Next Agent Runtime

Will not be defined by faster tool calls or cheaper tokens.

It will be defined by whether it can **preserve what has already become true**, **bound the authority that remains**, and **recover without re-entering the world blind**.

Experience: A decade building distributed execution systems at Cloudflare and Kong. What has changed is not just the workload, but what infrastructure is responsible for.

The Infrastructure Layer We Need

Semantic Continuity, Not Just Process Continuity

Effect Log, Capability Gateway, and Resumability are not optional features — they are **foundational primitives** for any agent that runs long enough to matter.

Effect Log

Record what became true

Capability Gateway

Bound remaining authority

Resumability

Recover without blind re-entry

 Distributed Systems |  Fault Tolerance |  Agent Infrastructure

The Infrastructure Layer We Need

Model will continuously generate entropy, The true mission of infrastructure is to reduce it.

The agent loop determines behavior.
Infrastructure determines the boundary.

The Infrastructure Layer We Need

Q&A