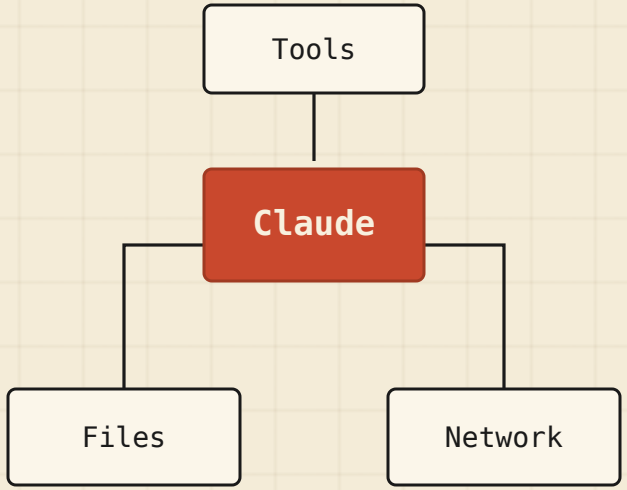


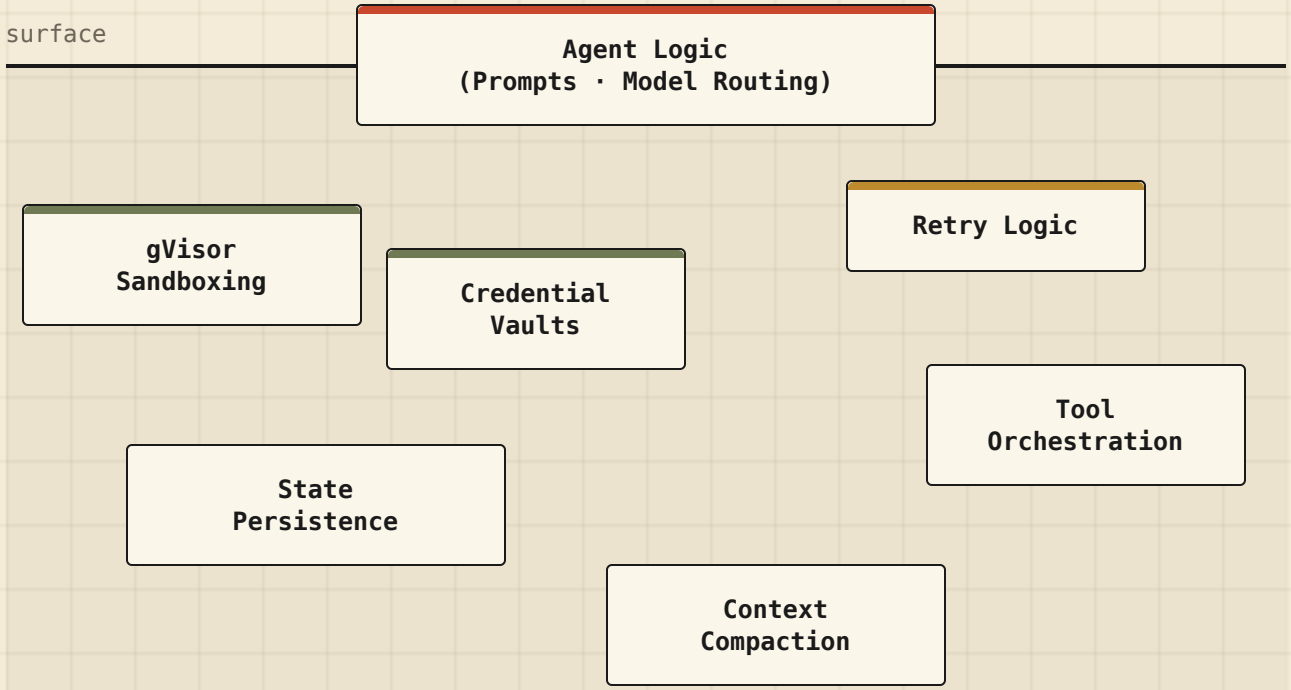
A PRODUCTION BLUEPRINT

The *Orchestration* Layer

A production blueprint for Claude Managed Agents.

Serverless Infrastructure · Enterprise Security · Multi-Agent Workflows

The infrastructure illusion



Insight

Building an agent is the easy part. **Running it reliably** is where teams get stuck.

Takeaway

Managed Agents abstract the execution layer. By handling sandboxing, auth, and state entirely server-side, engineering teams focus purely on business logic — not brittle agent loops.

Messages API vs. Managed Agents

	DIY: Messages API	Managed Agents
Session length	Short-to-medium	**Long-running (hours+)**
State persistence	Developer-managed	**Built-in (server-side, durable)**
Sandbox security	Developer-provisioned	**Fully managed (cloud or self-hosted)**
Advanced primitives	Not available	**Built-in (Outcomes, Dreaming)**
Batch API	Supported (50% discount)	**Not supported**
Compliance	HIPAA / ZDR eligible	**Not yet eligible (stateful by design)**

The core architecture triad

Block 1 — **The Harness**
Stateless orchestration loop · API calls

Block 2 — **The Session**
Durable, append-only event log

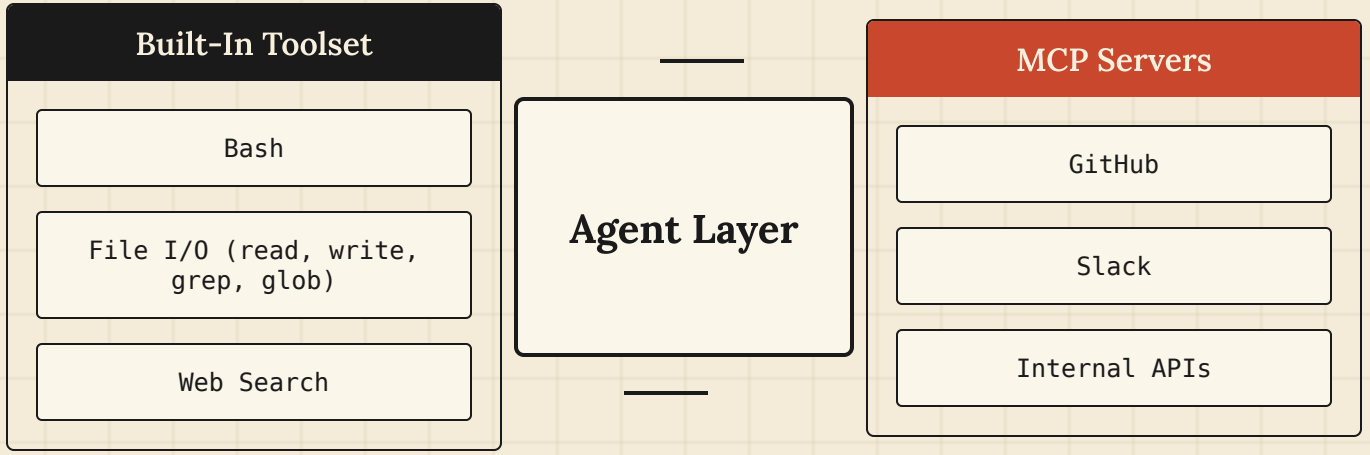
Block 3 — **The Sandbox**
Disposable gVisor container

Mechanics
Managed Agents separate execution into three decoupled components.

Resilience
If the Harness crashes, it resumes cleanly from the Session log.

Security
A compromised Sandbox cannot reach the Harness or alter the immutable event history.

Built-ins & MCP



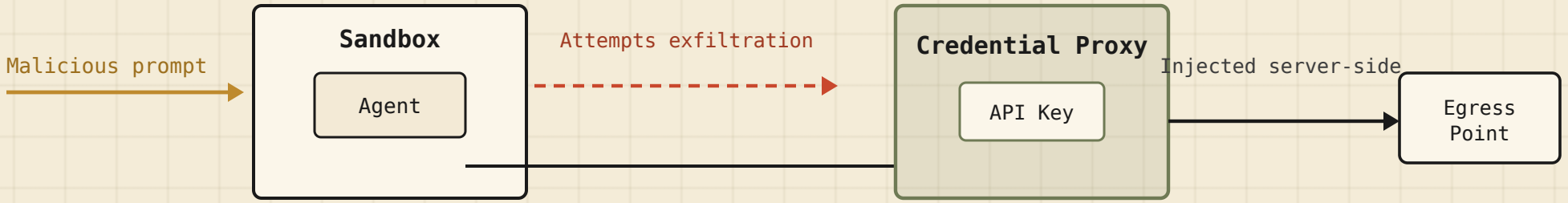
Concept

Claude autonomously selects and executes tools within the secure sandbox.

Extensibility

The Model Context Protocol (MCP) acts as a universal standard — a "USB-C for AI" — allowing standardised integration with third-party data sources and proprietary enterprise systems.

The vault system: structural credential protection



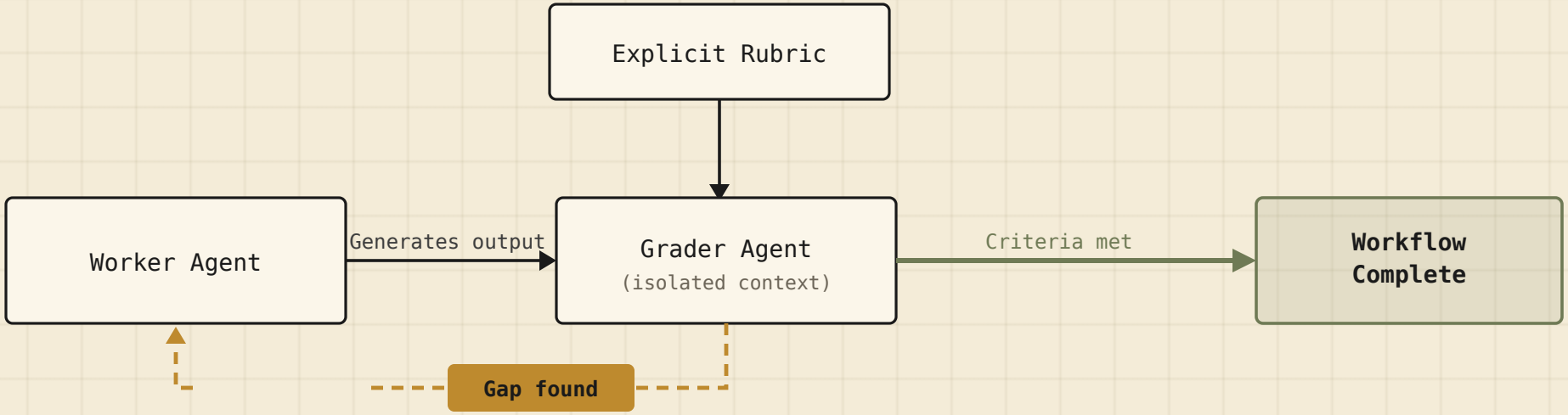
The Mechanism

Vault credentials never enter the gVisor sandbox address space.

The Guarantee

Even with a successful prompt injection, attackers cannot steal vault secrets. The attack surface shrinks from "steal any token" to "misuse existing tool permissions."

Defining ‘done’: the Outcomes primitive



The Problem

Standard agents stop when an output merely *appears* plausible.

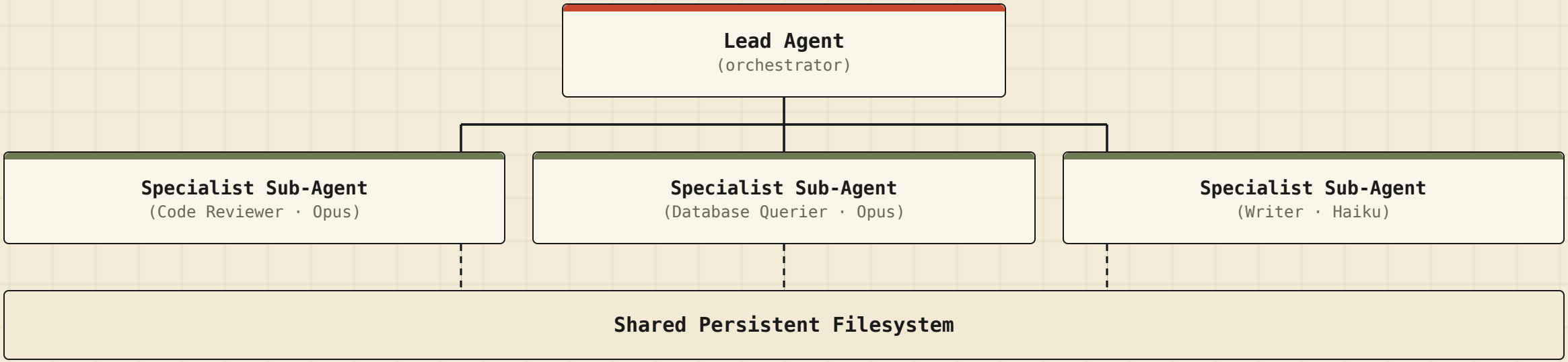
The Solution

A separate Grader Agent evaluates output against a rubric, looping until criteria are met or the iteration budget is spent.

The Result

Improves task success by up to 10 points on complex jobs (e.g., +8.4% on docx generation).

Multi-agent orchestration



Delegation

Tasks too large for one context window are routed by a Lead Agent to specialists.

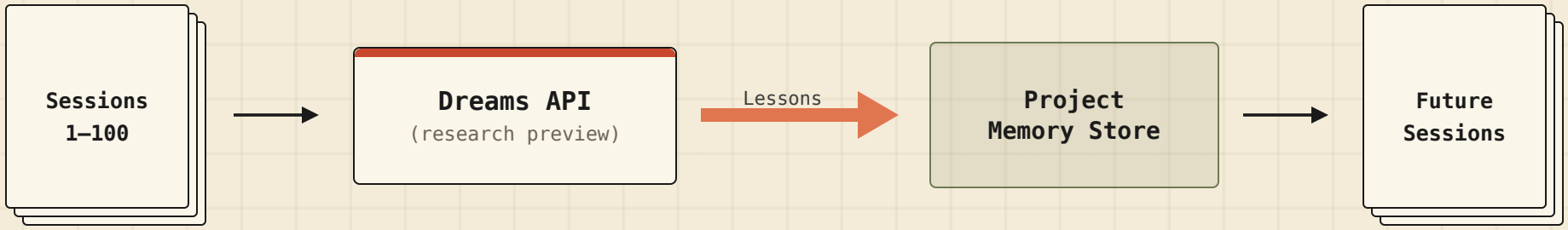
Isolation

Each sub-agent runs independently with distinct tools, models, and fresh context windows.

Collaboration

They communicate via event payloads and collaborate on a shared, persistent filesystem.

Memory consolidation: Dreaming



Concept

A formal mechanism for agents to learn between sessions — an automated post-mortem.

Mechanism

Non-parametric memory consolidation: analyses past runs, identifies successful tool patterns, merges duplicates, builds a structured runbook.

Impact

In early usage, Harvey reported a 6x improvement in completion rates as agents stopped repeating tool-specific mistakes.

The ‘defaults’ warning

QUICKSTART DEFAULTS

Auto-Execute (always_allow)

PRODUCTION HARDENED

Human-in-the-loop (always_ask)

The Risk

Out-of-the-box defaults are maximally permissive: full bash access, unrestricted outbound internet, zero human confirmation.

The Reality

Unrestricted networking + bash = a direct path to data exfiltration on prompt injection. Every deployment must be actively reconfigured.

The security hardening checklist

1. Limit networking: Restrict outbound traffic to an explicit allowed-hosts list.

```
"networking": {
  "type": "limited",
  "allowed_hosts": ["api.github.com"]
}
```

2. Least privilege: Start with tools disabled; enable only what is strictly required.

```
"default_config": {"enabled": false},
"configs": [{"name": "read", "enabled": true}]
```

3. Permission policies: Use always_ask for high-risk operations to force human confirmation.

```
"permission_policy": {
  "type": "always_ask"
}
```

4. Pin dependencies: Hardcode package versions in the environment config against supply-chain attacks.

```
"packages": {
  "pip": ["pandas==2.2.0"]
}
```

Understanding the economics

STANDARD MODEL TOKENS

Sonnet 4.6 · per MTok

$3	input vs $15 output

Prompt caching is built in natively, drastically reducing costs on long-running tasks.

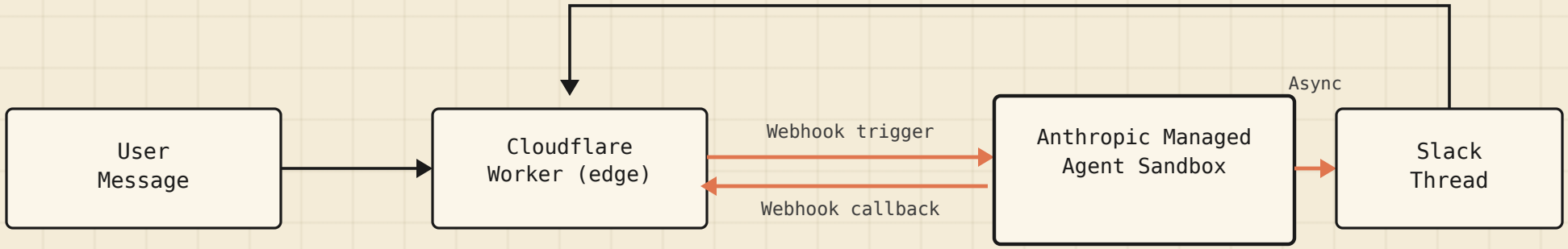
INFRASTRUCTURE FEES

Container runtime:
$0.08 / session-hour (active runtime only; idle is free).

Built-in web search:
$10 / 1,000 calls.

The Hidden Cost The real variable is tool-call token accumulation — every bash command and file read adds to the context window.

Production deployment patterns



The Pattern

The ideal production setup bypasses dedicated backend polling services.

The Implementation

Use serverless edge layers (Cloudflare Workers) to fan out tasks and handle webhook callbacks.

The Result

Server-Sent Events relay real-time progress while the container filesystem persists across async follow-ups.

The production AI stack

The Paradigm Shift

Everyone else built a construction worker. Anthropic built the **contractor**.

The Conclusion

You provide the success rubric, the secure environment, and the tools. Managed Agents handle the orchestration, the security boundaries, and the state management.

Next steps for engineering teams

> Phase 1: Prototype

Use the Claude Console to design the base Agent configuration and Environment YAML.

> Phase 2: Secure

Audit the configuration. Migrate all credentials to Vaults and enable Limited Networking.

> Phase 3: Deploy

Implement via the SDK. Integrate Webhooks and the Outcomes primitive for asynchronous reliability. Ensure the required header is active: `managed-agents-2026-04-01`