sentrely

THE CONTROL PLANE PLAYBOOK

Deploy AI agents with confidence.

The practical playbook for governing autonomous AI coding & ops agents — Claude Code and Codex — in production, without handing them the keys to everything.

WHAT'S INSIDE

From "move fast" to "move fast, safely."

AI coding agents went from autocomplete to autonomous operators in under two years. They read your repos, push code, touch AWS, run migrations, and call external tools — often on the same credentials their human operator holds. Powerful. Also ungoverned by default.

This playbook is the field guide we wish existed: the specific, deterministic controls that let you run agents at full speed **and** answer the question every security team eventually asks — **"what did the agent actually do, and what could it have done?"**

90%
of a runaway agent's spend goes to context, retries & overhead — not real reasoning

$10k+
/mo in metered API tokens for teams running agents all day

1
control plane to deploy, govern, observe & orchestrate them all

You'll walk away with

- The 5 practices that turn an ungoverned agent into a governed one
- The control-plane model: Deploy · Orchestrate · Govern · Observe
- How to stop renting tokens and cut agent cost without losing capability
- A copy-paste implementation checklist

CHAPTER 1

The oversight gap

Run Claude Code or Codex with `--dangerously-skip-permissions` and the agent inherits the full access of whoever launched it — AWS, prod databases, deploy keys, the lot. It works beautifully right up until the moment it doesn't: an agent "cleaning up unused infrastructure" that runs `terraform destroy` on the wrong workspace, or a migration script pointed at the wrong database.

The instinct is to fix this with a better prompt — "never touch production," "always ask first." But a prompt is a suggestion, not a control. The model can misread it, an injection can override it, or the agent can simply decide the rule doesn't apply this time.

> *"LLMs are probabilistic reasoning engines, not security enforcement mechanisms."*
>
> — AWS Security Blog, 2026

That single sentence is the whole thesis. You don't make an agent safe by asking it nicely. You make it safe by putting **deterministic controls outside the agent** — in a layer it cannot edit, skip, or talk its way around. The rest of this playbook is how.

The three questions a governed setup can answer

- **Identity:** which agent did this, under whose authority?
- **Authorization:** was it allowed — and how do we know it couldn't do more?
- **Audit:** can we replay exactly what happened, after the fact, tamper-free?

Earn confidence, one control at a time

1 Give each agent its own identity & least privilege

Don't run agents on a human's admin credentials. Issue a dedicated machine identity scoped to exactly the actions and resources the job needs — `s3:GetObject` on one bucket, not `s3:*`. Differentiate agent-initiated actions from human ones and hold the agent to a tighter standard. If the identity is unique, every later control — authorization, audit, cost — has something real to attach to.

2 Authorize every action downstream, deterministically

OWASP calls this **complete mediation**: the decision "is this allowed?" must live in a layer the agent can't bypass — not in the agent's own config (editable, skippable) and not in the prompt (advisory). Default to deny: nothing runs unless a policy explicitly permits it. This is the difference between "we told it not to" and "it cannot."

> *A denylist asks "is this one of the bad things?" An allowlist asks "is this one of the few good things?" Only the second is safe when the attacker — or the model — is creative.*

3 Gate high-impact actions on a human

Reading a file is reversible. `terraform destroy`, dropping a table, rotating a key, or emailing 10,000 customers is not. Require human approval for the risky few percent — routed to where people already work (Slack, Telegram) — and **only** those. Approve everything and the approval becomes a rubber stamp; approve the genuinely dangerous and each one stays meaningful.

4 Log everything, immutably

Your provider's usage dashboard shows tokens consumed; it does not show the commands run. You need a tamper-proof, per-action record — tied to a specific agent identity and session, captured at the gateway, independent of the agent process — so a crashed, confused, or compromised agent can't erase its own trail. This is what turns "we think it only read data" into "here is the signed log of every call."

Every

tool call captured — args, result, decision, approver

Immutable

append-only, legal-grade, exportable for SOC 2

5 Put a control plane between agents and infrastructure

The first four practices are painful to bolt onto each agent one at a time — and impossible to trust when the enforcement lives inside the thing being enforced. So route **all** agent traffic through one external layer. The agent holds zero credentials: it asks the gateway, the gateway checks policy, the gateway acts and logs. Identity, authorization, approvals, audit, and cost controls all live in one place you actually control.

It's also the only natural home for the per-tool-call checkpoint that the Model Context Protocol (MCP) itself doesn't define. MCP standardized how agents reach tools; it left "should this specific call be allowed right now?" as an exercise for the reader. The control plane is that exercise, solved once, for every agent.

> *Governance you bolt onto an agent is a setting. Governance the agent must route through is a guarantee.*

That shift — from setting to guarantee — is the entire reason a control plane exists.

CHAPTER 3 · THE MODEL

One control plane, four jobs

Everything an agent does flows through one managed gateway — so autonomy never means losing oversight. Four responsibilities, one surface:

DEPLOY

Run Claude Code & Codex agents inside one managed, contained environment. Agents hold zero credentials and never touch your infrastructure directly.

ORCHESTRATE

Coordinate multi-agent workflows — agent-to-agent messaging, live session tracking, and routing — from a single pane.

GOVERN

Deny-by-default policy per agent, least-privilege scoped to action + resource, and human approval on risky operations.

OBSERVE

An immutable, per-tool-call audit trail and a real-time dashboard — so you always know exactly what every agent did, and why.

If you only remember one diagram from this playbook, remember this one. Deploy puts the agent in a box; Govern decides what the box allows; Observe records it; Orchestrate lets many boxes work together.

CHAPTER 4 · ECONOMICS

Stop renting tokens by the hour

Optimization — shorter sessions, scoped context, hard budget stops — trims a metered bill. But the question worth asking first is blunter:

BRING YOUR OWN PLAN

Who has $10k/mo to burn on agent tokens?

Run your agents on the Claude Max or ChatGPT / Codex subscription you already pay for — connected via OAuth — instead of metered API tokens that rack up all day. Bring your plan, not your token bill: flat-rate horsepower, no surprise invoice.

The shape of the bill

Same agents, same horsepower — a flat subscription instead of a meter that punishes you for using the tool you bought. Sentrely supports both: an API key when you want raw per-token control, or BYO subscription via OAuth when you want a predictable number.

CHAPTER 5 · DO THIS

The implementation checklist

- [ ] **Give every agent a unique identity**
  No shared human creds. One identity per agent, scoped to its job.
- [ ] **Switch to deny-by-default authorization**
  Allowlist the few good actions; everything else is blocked at the gateway.
- [ ] **Define your "requires approval" list**
  Destroy, drop, delete, rotate, mass-send, prod writes — gate these on a human.
- [ ] **Route approvals where people already are**
  Slack or Telegram, with full context, so decisions take seconds not hours.
- [ ] **Turn on immutable, per-call audit**
  Captured at the gateway, tamper-proof, exportable for compliance.
- [ ] **Set per-agent cost controls**
  Hard budget stops, not soft warnings. Agents don't read warnings.
- [ ] **Move heavy agents to a flat plan**
  BYO Claude Max / Codex via OAuth instead of a metered API meter.
- [ ] **Put it all behind one control plane**
  So enforcement lives outside the agent — a guarantee, not a setting.

sentrely

GET STARTED

Deploy AI agents with confidence.

Sentrely is a fully-managed control plane for Claude Code & Codex agents — policy, approvals, and audit, with no infrastructure to run, and bring-your-own-plan so you're not renting tokens.

Set up in minutes | From $199/mo | BYO Max / Codex plan | Deny-by-default + audit

Start free → sentrely.com

Questions? Email jordan@sentrely.com