

The Principles on One Page

Twelve principles and the decisions behind them, for anyone who designs, ships or runs systems built on Claude.

v1.0 · October 2026

Daniel Wanja

THE TWELVE PRINCIPLES

1 Start from the outcome, not the model	2 Verify it or contain it	3 Use the simplest shape that works	4 Fit the model with evidence; spend context deliberately
5 Grant the least capability; authorize every hop	6 Send each question where its answer lives	7 Define “good” before you build	8 Change one thing and measure it
9 Attack the dominant cost; trace every hop	10 Enforce, don’t request	11 Say it out loud, and write it down	12 Configure the team, not just the tool

FRAME Start from the outcome Ch 1

- **Problem statement:** “For [actor], when [unit] arrives, produce [artifact] so [metric] moves [baseline] → [target].”
- **Split the work:** code does rules and arithmetic; Claude does messy judgment.
- **One value pillar leads** (efficiency, transformation, productivity, cost, performance); the rest are guardrails.
- **Discovery:** map who can veto; read 50 real documents before you design.
- **Verdicts:** Go · Go with conditions · Not yet · Not with LLMs · No.
- **Price the outcome:** model cost × attempts + review time + escaped errors.

SHAPE Use the simplest shape that works Ch 2

Climb on evidence: Single call → Augmented model → Workflow →

Orchestrator-workers → Agent

Workflow pattern	Use it when
Prompt chaining	Steps are known and fixed
Routing	Inputs fall into distinct categories
Parallelization	Parts are independent, or votes help
Orchestrator-workers	Subtasks can’t be known in advance
Evaluator-optimizer	Criteria are clear; feedback helps

- **Workflow:** your code picks the next step. **Agent:** the model does, so add stop limits and checkpoints.

CONFIGURE Pick the model, spend context Ch 3

- **Tiers:** Haiku for speed and volume · Sonnet for balance · Opus and above for the hardest work.
- **Choose by the hardest common case.** Unsure? Start on Opus; step down while evals pass.
- **Compare cost per completed task.** Tune effort before you switch models.
- **Fewest high-signal tokens wins.** Long documents first, question last.
- **Cache order:** tools → system → messages; stable first, no timestamps up top.

CONNECT Give agents less, connect carefully Ch 4

Control order: Remove → Narrow → Gate → Detect

Situation	Connect with
Another team’s agent	Agent-to-agent (A2A)
Reused by several apps	An MCP server
Scripts in a sandbox	CLI or code execution
One tool, one app	A direct tool definition

- **Prompt = hint; runtime = boundary.** Never pass a user’s token through.
- **Never mix** private data, untrusted content and an outbound channel.
- **Load on demand** (tool search, Skills); always load must-apply policy.

GROUND Send each question where its answer lives Ch 5

The question needs	Send it to
Facts in prose	Hybrid search + reranking
Totals, counts, joins	A query tool (SQL), not search
Current state	A live call to the owning system
A small corpus (≈500 pages)	The whole corpus, cached

- **IDs and codes** → keyword · **paraphrases** → semantic · **both** → hybrid.
- **Contextual Retrieval:** 35% fewer failed retrievals; 49% with keyword search; 67% with reranking.
- **Filter by permission before ranking.** Every answer cites its source.

PROVE Define “good” before you build Ch 6

- **Every criterion** = metric + threshold + test set + grader.
- **Measure:** accuracy · latency · cost · safety · security. Gate severe errors separately.
- **Test set:** golden + edge + adversarial + production cases; keep a held-out set.

If the output is	Grade it with
Checkable	Code
Nuanced (tone, completeness)	A model judge, calibrated to humans
Unlabeled or contested	Human reviewers

Roll out: Offline eval → Shadow → Canary → A/B test

- **One change at a time.** At 80% pass: 50 cases ≈ ±11 pts; 400 ≈ ±4.

OPERATE Make cost and speed visible Ch 7

Dominant cost	Lever
Long prefix on every call	Prompt caching
Nobody is waiting	Batch: half price, within 24 h
Many easy requests	Route to a smaller model
Long outputs	Cap length; ask for structure
Users feel the wait	Stream (perceived speed only)

- **Attack the largest term first.** Split the latency budget by stage; track p50 and p95.
- **One trace per request.** Log structure always, content rarely, secrets never.

GOVERN Enforce, don’t request Ch 8

- **Ignore test:** if the model ignored this rule, what would still stop the harm?
- **Prompts steer; code enforces.** Structured outputs fix shape, not truth.

Stakes	Human role
Low	Automate; sample for quality
Medium	Review when the system is unsure
High or irreversible	A person approves first

- **Approval gate:** exact action shown, bound to its payload, silence = no, logged.
- **Compliance:** map each rule to a control and an owner.
- **Fairness:** report per group; a 95% average can hide 75%.

ALIGN Say it out loud, write it down Ch 9

They ask for	Agree on
“100% accurate”	Test-set threshold + human review
“Instant”	p50/p95 at the client + first token
“Always on”	Internal SLO; looser SLA + fallbacks

- **Options:** 2–4, one without a model. Decision records are superseded, never edited.
- **Hand over:** architecture, decisions, prompts, evals, runbooks, known limits.
- **Models retire** with ≥60 days’ notice: test new ones as they ship.

ENABLE Configure the team, not just the tool Ch 10

- **CLAUDE.md informs** (under 200 lines); settings, permissions and hooks **enforce**.
- **Permissions:** deny → ask → allow; a deny anywhere wins. Hooks only tighten.
- **Precedence:** org-managed > CLI > local project > shared project > user.
- **Approve MCP servers centrally**, by address, not name. Sandbox the shell.

Debug: Replay trace → Isolate layer → Fix → Add regression test