AI AGENT ENGINEERING SERIES • V1.0 EDITION

Mastering Subagents & Orchestration

The practical visual field guide to designing, permission-scoping, and orchestrating specialized AI coding teams—from single context windows to parallel worktree fleets and adversarial review loops.

Context Isolation Architecture

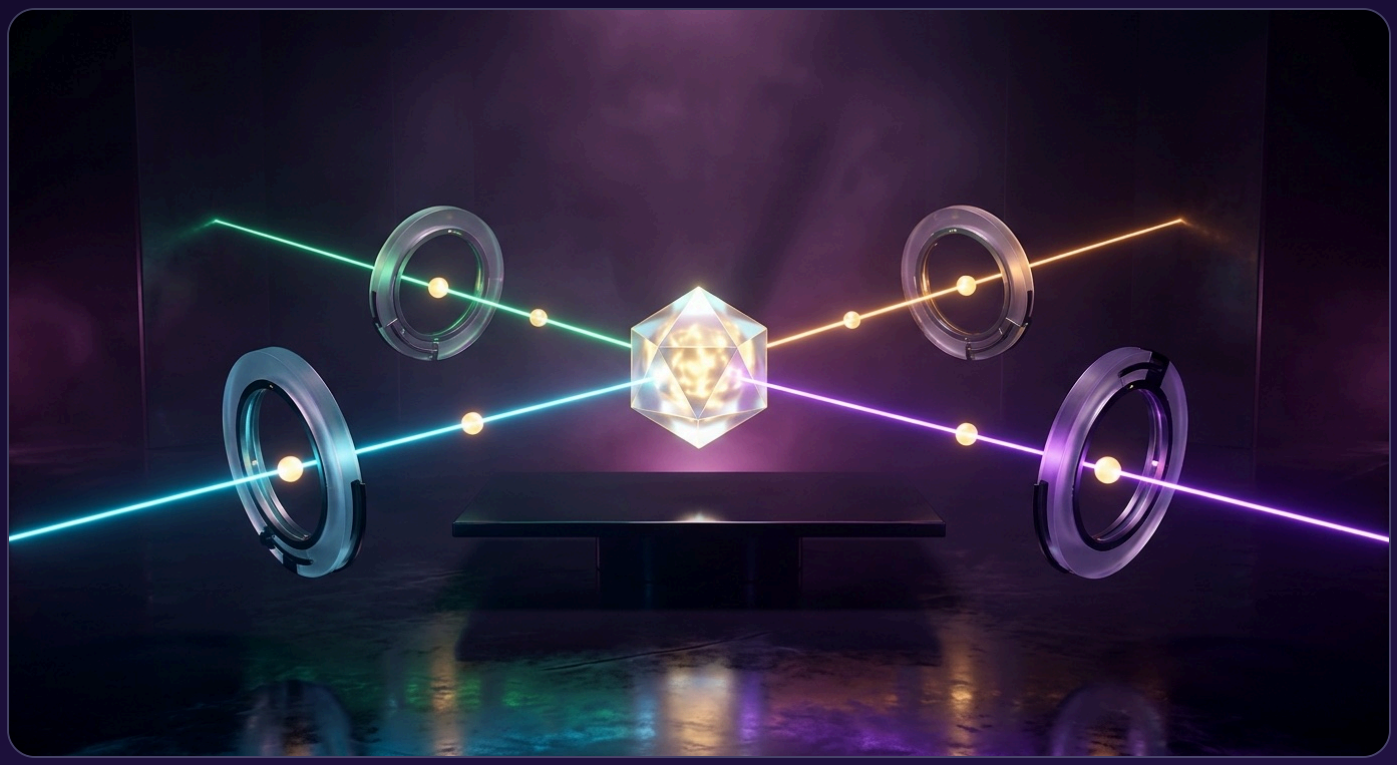
`.claude/agents/` & YAML Config

Forks vs. Clean-Slate Workers

Least-Privilege & `PreToolUse` Hooks

Fan-Out, Worktrees & Critic Loops

Claude • Antigravity • Cursor • Codex



Protect Your Main Context

Keep 80,000+ tokens of noisy file searches and test logs out of your lead session—only crisp summaries return.

Config-Enforced Safety

Prompts persuade, but configuration forbids. Lock scouts to read-only tools and gate shell calls with hooks.

Portable Across Ecosystems

Master one blueprint that maps cleanly across Claude Code, Google Antigravity, Cursor, Copilot, and Codex.

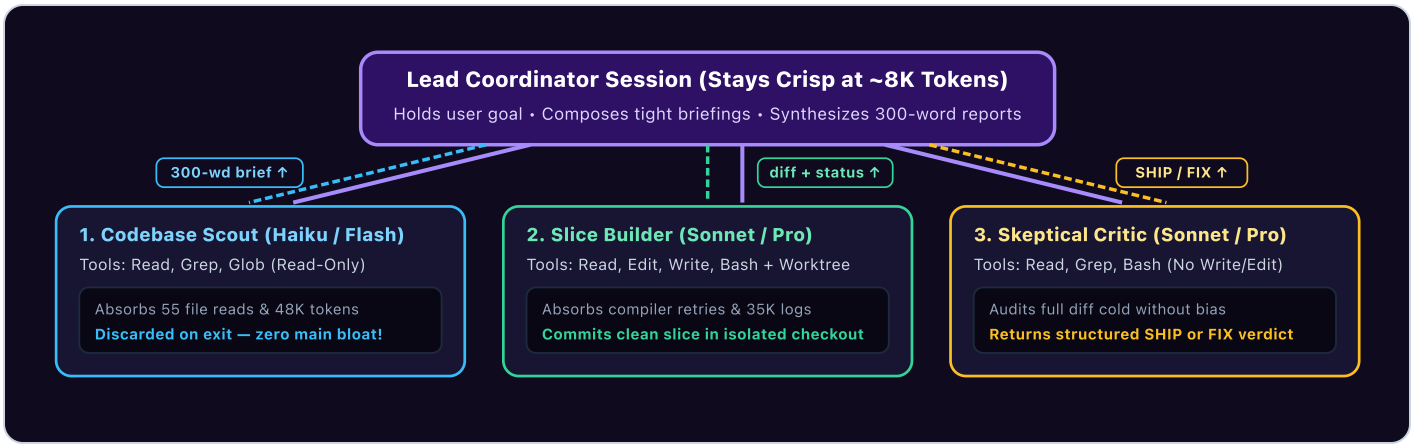
Why Context Isolation Beats One Giant Chat Thread

A **subagent** is a focused specialist spawned by your main coding agent to execute one bounded job inside **its own private context window**, governed by **its own system prompt, model, and tool permissions**, returning only a distilled final report.

Quick Navigation

1. Context Isolation Mental Model p. 2
2. Four Architectures & Decision Matrix p. 3
3. Live Dispatch & Config UI Tour p. 4
4. Visual Topologies & YAML Reference p. 5
5. Three Production Agent Blueprints p. 6
6. Pre-Flight Checklist & Clinic p. 12
7. Invocation, Forks & Concurrency p. 7
8. Least-Privilege & Security Hooks p. 8
9. Five Copy-Paste Orchestration Recipes p. 9
10. Cross-Vendor Ecosystem Mapping p. 10
11. End-to-End Pipeline & Cost Math p. 11

The Context Economy: Expensive Exploration Dies Locally, Cheap Truth Flows Up



The Law-Firm Analogy That Explains Every Design Rule in This Guide

Think of your main session as a **senior partner** and every subagent as a **brilliant junior associate with total amnesia**. A fresh subagent knows *nothing* about your chat history except the briefing paragraph you pass it, and can only touch the exact tools on its allowlist. It reads 400 pages of discovery in a back room and hands you a crisp one-page memo—keeping your main desk clear for high-judgment decisions.

Four Multi-Agent Topologies & When to Delegate

"Multi-agent coding" actually describes four distinct operating patterns. Picking the right pattern before you write a prompt saves both wall-clock time and 3x–5x in wasted token spend.

1 Single Agent (Inline Default)

One context window, one tool loop. Best for 30-second targeted edits, interactive steering, and sequential steps that all rely on the exact same files already in memory.

2 Focused Subagent (Hub-and-Spoke)

Coordinator spawns clean-slate specialists (`researcher`, `test-writer`, `reviewer`) via sequential pipelines, parallel fan-out, or intent routing, then merges their summaries.

3 Context Fork (`/subtask` • `fork`)

Clones your *entire active conversation* into a background branch sharing your prompt cache. Ideal when a side task needs everything discussed so far without polluting your main thread.

4 Peer Swarm / Agent Team

Multiple long-lived peer sessions collaborating across a shared message board or task roster (Antigravity `/teamwork` Swarms, Claude Agent Teams) on multi-hour parallel builds.

The Money-Saving Decision Matrix: Inline vs. Subagent vs. Fork vs. Swarm

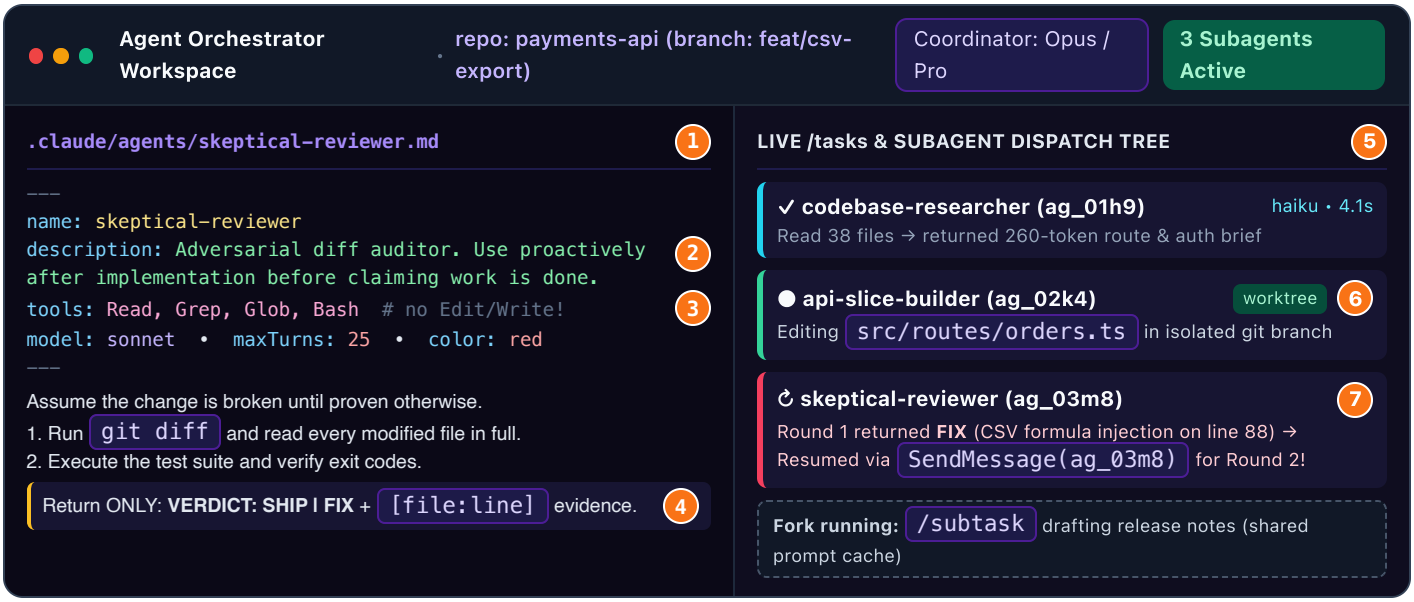
Task Signal	Do Inline (Main)	Focused Subagent	Context Fork	Peer Swarm / Team
Output Volume	Small code diff you will keep editing	Floods of grep hits, stack traces, or test logs	Side artifact (PR notes, unit test draft)	Multiple full subsystems + integration checks
Context Needed	Already loaded in current chat	Fits cleanly in a 4-sentence briefing	Needs 100% of current chat decisions	Shared design spec + live peer board updates
Back-and-Forth	High interactive human steering	Self-contained input → report contract	Fire-and-forget from current state	Agents negotiate API contracts laterally
Safety Posture	Uses your session's full toolset	Strict least-privilege (e.g., read-only)	Inherits parent tools & prompt cache	Per-worker git worktrees + dedicated verifier
Token Cost	**1x** (lowest baseline)	**1.2x–3x** (cheap model offset)	**Low first turn** (shared cache hit!)	**4x–8x** (highest; reserve for big scope)

What Subagents Are NOT: 4 Myths That Burn Budgets

1. Not automatic edit safety: Two workers editing the same file concurrently will overwrite each other unless you partition folders or enable `isolation: worktree`. • **2. Not a free cache:** Every fresh spawn re-reads files unless you *resume* its Agent ID. • **3. Not zero-cost:** Five parallel agents spend ~5x the tokens. • **4. Not a replacement for Skills:** If a reusable workflow needs main-chat context (like formatting a commit), make it a `SKILL.md`, not a subagent.

Annotated Tour: Subagent Spec & Live Dispatch Monitor

Every modern subagent system pairs a declarative **definition file** (left) with a live **task & transcript monitor** (right) so you can inspect active models, tool boundaries, and isolated worktrees at a glance.



Numbered Architecture & Control Reference

1 **Version-Controlled Agent File:** Checked into `.claude/agents/` (also auto-detected by Cursor and VS Code!) so your entire engineering team shares identical specialists.

2 **Classifier `description`:** The orchestrator reads this string to decide *when* to auto-delegate. Include trigger phrases like *"Use proactively after..."*.

3 **Explicit Tool Allowlist:** Omitting `Edit` and `Write` guarantees the reviewer cannot "helpfully" rewrite broken code while auditing it.

4 **Machine-Carriable Output Contract:** Forcing a predictable `VERDICT: SHIP | FIX` structure lets the coordinator branch cleanly without guessing.

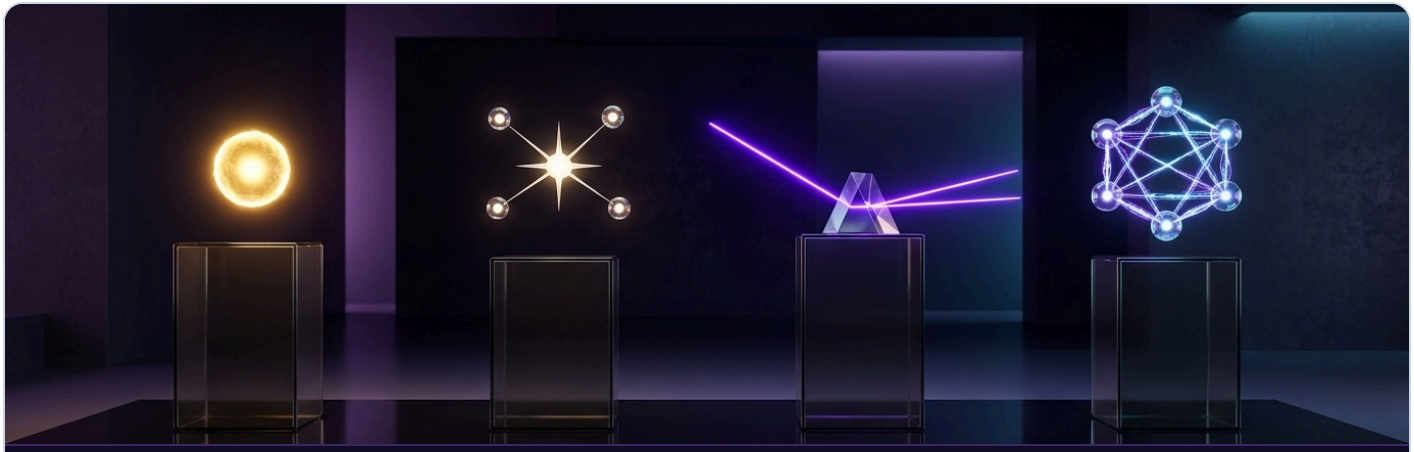
5 **Live `/tasks` Panel:** Displays every running and recent subagent with its active model, effort level, turn count, and color badge; press `Enter` to inspect its raw log.

6 **Worktree Isolation Badge:** Confirms parallel builders are writing to separate temporary git checkouts so file edits never collide.

7 **Agent ID & Stateful Resume:** Every completed worker returns a handle (like `ag_03m8`). Messaging that ID resumes its full memory for Round 2 instead of paying to re-read the repository from scratch.

The 4 Topologies & Subagent Frontmatter Reference

A subagent file is plain Markdown: YAML frontmatter between `---` fences on top, and the specialist's system prompt below. Only `name` and `description` are mandatory—every other key sharpens cost, isolation, or safety.



1. SINGLE AGENT: One shared context & tool loop. | **2. HUB & SPOKE:** Coordinator dispatches scoped workers. | **3. CONTEXT FORK:** Splits live thread w/ shared cache. | **4. PEER SWARM:** Lateral mesh on shared task board.

Complete Subagent YAML Frontmatter Reference

Field	Req?	Behavior, Precedence & Practical Guidance
`name`	**Yes**	Unique lowercase-hyphen ID (e.g., `codebase-researcher`). Avoid colons (reserved for plugin namespaces).
`description`	**Yes**	Natural-language classifier trigger telling the coordinator *when* to delegate (all agents share a 15K-token startup budget).
`tools`	No	Explicit allowlist (`Read, Grep, Glob, Bash`). **Warning:** omitting `tools` inherits every session & MCP tool!
`disallowedTools`	No	Denylist subtracted *before* `tools` resolves (e.g., `Write, Edit, Agent, mcp__github`).
`model`	No	`haiku`, `sonnet`, `opus`, full ID, or `inherit`. Precedence: per-call → frontmatter → `CLAUDE_CODE_SUBAGENT_MODEL` → parent.
`permissionMode`	No	`default`, `acceptEdits`, `auto`, `dontAsk` (pre-allowed only), `plan` (read-only), or `bypassPermissions` (CI containers only).
`hooks`	No	Agent-scoped lifecycle scripts (`PreToolUse`, `PostToolUse`, `Stop`) to validate CLI arguments or auto-format edits.
`isolation` · `maxTurns` · `memory`	No	`isolation: worktree` prevents parallel edit collisions; `maxTurns` caps runaway spend; `memory: project` persists notes.

Pre-Flight Validation Tip: Catch YAML Typos Before Starting a Session

Broken YAML or missing `name` / `description` keys cause silent skips at startup. Run `claude plugin validate .claude/agents` (and `/doctor` to spot duplicate names) right after editing!

Prompt Anatomy & 3 Production-Grade Subagents

You do not need twenty exotic agents. Ninety percent of real-world engineering teams get maximum leverage from just three tightly scoped specialists: a **Scout**, a **Test-Runner**, and a **Skeptical Critic**.

The 5-Part Subagent Specification Formula (Color-Coded)

[1. ROUTING DESCRIPTION] "Read-only codebase scout. Use proactively before planning or multi-file edits to locate symbols & conventions." [2. LEAST-PRIVILEGE TOOLS] tools: Read, Grep, Glob • model: haiku (fast, cheap, zero write risk). [3. ROLE & ANTI-BIAS STANCE] "You never modify files or propose speculative rewrites—report verified repository facts only." [4. STEPWISE SOP] "1. Glob target directories. 2. Grep call sites. 3. Read entry points and existing test harnesses in full." [5. STRICT OUTPUT CONTRACT] "Return ONLY four headings: Findings (with file:line), Key Files, Conventions, and Open Questions."

Three Battle-Tested Agents Ready for `.claude/agents/` (or Any Repo)

1. `codebase-researcher`

Posture: Read-only scout on a fast, low-cost model.

```
---
name: codebase-researcher
description: Read-only repo
  investigator. Use before
  planning or implementation.
tools: Read, Grep, Glob
model: haiku
color: cyan
---
Never edit files. Map exact
entry points & conventions.
Return ONLY:
## Findings (file:line)
## Key Files
## Conventions Observed
## Open Questions
```

2. `test-writer`

Posture: Edit-capable, scoped strictly to test suites.

```
---
name: test-writer
description: Writes & runs unit
  and route tests after code
  changes. Use proactively.
tools: Read, Write, Edit,
  Grep, Glob, Bash
model: sonnet
---
1. Match existing fixtures &
   naming in tests/.
2. Write failing cases first,
   then run suite until green.
3. Never alter prod src/ code;
   report prod bugs upstream!
```

3. `skeptical-reviewer`

Posture: Blind adversarial verifier; returns verdict only.

```
---
name: skeptical-reviewer
description: Adversarial diff
  reviewer. Use before telling
  the user work is complete.
tools: Read, Grep, Glob, Bash
model: sonnet
maxTurns: 25
---
Assume changes are broken until
proven. Read full changed files
& run tests yourself. Return:
## VERDICT: SHIP | FIX
## Blocking [file:line]
## Verification Evidence
```

Why Separating the Implementer from the Reviewer Changes Everything

When the same agent writes a feature and checks its own work in one conversation, it suffers from **confirmation bias**—anchored by its own earlier rationalizations. Spawning `skeptical-reviewer` in a clean context forces cold, independent verification of the actual `git diff` and exit codes!

Invocation Mechanics, Context Forks & Concurrency

Understanding what crosses the context boundary at launch—and when to reach for a **fork** or **resume** instead of a fresh spawn—eliminates the "amnesia tax" between pipeline stages.

Four Ways to Trigger a Subagent (From Soft Routing to Guaranteed Pinning)

Trigger Mechanism	Syntax Example	Guarantee & Under-the-Hood Behavior
1. Automatic Delegation	*"Audit the auth module and summarize risks."*	Coordinator matches task intent against loaded `description` strings and calls the `Agent` tool automatically.
2. Natural-Language Name	*"Use the skeptical-reviewer agent on my diff."*	Very high likelihood; coordinator still composes the task briefing passed across the boundary.
3. Explicit `@` Mention	`@agent-skeptical-reviewer check HEAD~1`	**Guaranteed execution** of that exact subagent. Note: your coordinator still writes the subagent's briefing prompt!
4. Whole-Session Mode	`claude --agent skeptical-reviewer`	Runs your *main thread itself* under that agent's system prompt, tool allowlist, and model from turn one.

Clean-Slate Subagent vs. Context Fork (`/subtask` • `fork`)

A Clean-Slate Subagent (Default)

- **Sees at startup:** Its own Markdown prompt, the coordinator's briefing, repo `CLAUDE.md` / `AGENTS.md`, git status snapshot, and preloaded `skills`.
- **Watch out:** Built-in lightweight scouts (like `Explore`) skip `CLAUDE.md` to save tokens—restate critical folder exclusions directly in your prompt!

B Context Fork (`/subtask` or `fork`)

- **Sees at startup:** Your *complete parent conversation history*, tools, and system prompt—while keeping all new tool calls isolated in the background!
- **Cache superpower:** Shares the parent's **prompt cache**, making its first turn dramatically cheaper than re-briefing 40K tokens of context from scratch.

Three Runtime Rules Every Orchestrator Should Know

1 Resume, Don't Respawn

Completed agents retain an **Agent ID**. Asking the coordinator to *"resume that same reviewer"* sends a follow-up via `SendMessage` with full history intact —zero re-reading!

2 Nesting Depth Cap (3)

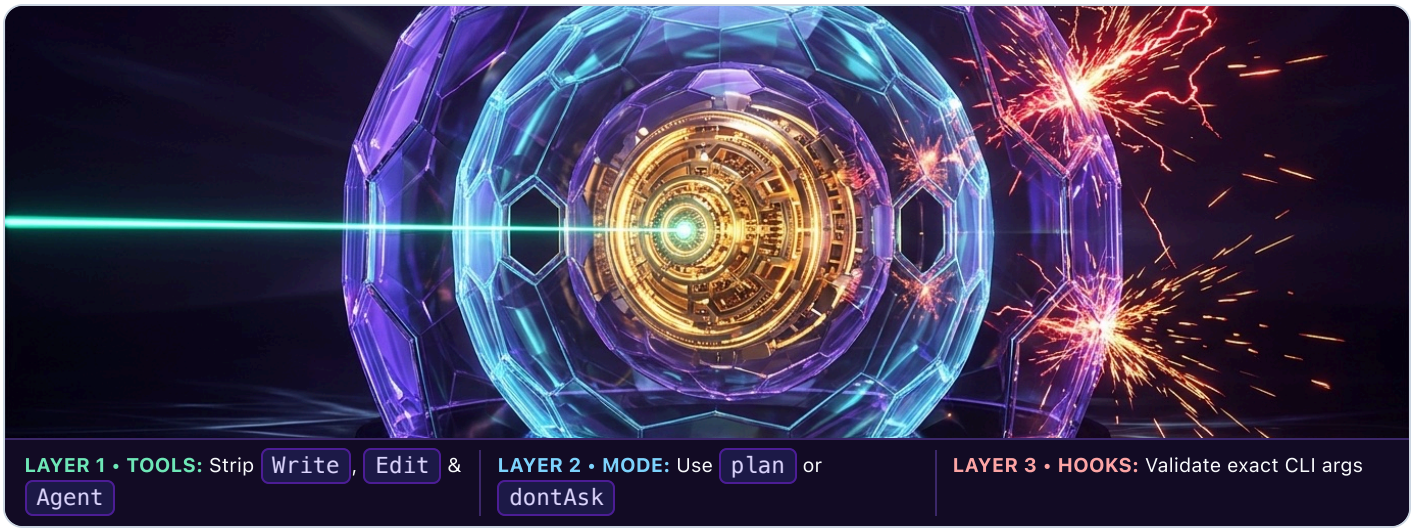
Subagents can spawn child helpers up to **3 layers deep** by default (tuned via `..._MAX_SUBAGENT_SPAWN_DEPTH`; set `1` for flat hub-and-spoke). Forks never re-fork.

3 Concurrency Ceiling (20)

Up to **20 background subagents** run simultaneously before queueing (tuned via `..._MAX_CONCURRENT_SUBAGENTS`). Press `Ctrl+B` anytime to background a live run.

Least-Privilege Scoping & Command-Inspecting Hooks

Once subagents run unattended in the background—often reading untrusted web pages, GitHub issues, or third-party READMEs—prompt politeness is not a security boundary. Enforce blast-radius limits in configuration.



LAYER 1 • TOOLS: Strip `Write`, `Edit` & `Agent`

LAYER 2 • MODE: Use `plan` or `dontAsk`

LAYER 3 • HOOKS: Validate exact CLI args

Layer 1 & 2: Tool Filtering Dials & Per-Agent Permission Modes

Allowlist vs. Denylist + Spawn Locking

- **Narrow job? Use `tools`:** `tools: Read, Grep, Glob` guarantees zero file writes and zero shell access.
- **Broad job? Use `disallowedTools`:** Subtract dangerous capabilities like `Write, Edit, mcp__github`.
- **Restrict who spawns whom:** A main coordinator with `tools: Agent(researcher, test-writer), Read` is structurally blocked from spawning any unlisted editor!

Choosing `permissionMode` Safely

- `plan`: Enforces structural read-only mode inside scouts.
- `acceptEdits`: Smooth local coding inside isolated worktrees.
- `dontAsk`: Auto-denies anything outside pre-approved rules—ideal for unattended background runs.
- **Note:** If your parent session runs `bypassPermissions`, children inherit it—reserve bypass strictly for disposable containers!

Layer 3: Argument-Level Guardrails with a `PreToolUse` Bash Hook

Giving an analyst `tools: Bash` lets it run `psql`—but also `rm -rf`! Pair frontmatter `hooks` with a 10-line validator script where `exit 2` blocks bad commands and feeds `stderr` straight back to the subagent:

```
# .claude/agents/db-analyst.md
---
name: db-analyst
description: Read-only SQL reporting analyst.
tools: Bash, Read
hooks:
  PreToolUse:
    - matcher: "Bash"
      hooks:
        - type: command
          command: "./scripts/readonly-sql.sh"
---
```

```
#!/usr/bin/env bash
# ./scripts/readonly-sql.sh (remember: chmod +x!)
CMD=$(jq -r '.tool_input.command // empty')
if echo "$CMD" | grep -qiE \
  '\b(INSERT|UPDATE|DELETE|DROP|ALTER|TRUNCATE|rm)\b';
then
  echo "Blocked: SELECT queries only." >&2
  exit 2   # Hard-blocks call & informs agent!
fi
exit 0
```

5 Copy-Paste Orchestration Recipes That Pay Off Daily

Drop these five battle-tested prompts straight into your coordinator session. Each combines explicit boundaries, isolation, and convergence rules so agents finish cleanly without babysitting.

Recipe & Pattern	Agents & Isolation Setup	Copy-Paste Kickoff Prompt
1. Sharded Repo Security & Architecture Audit Parallel Fan-Out → Merge	3x `codebase-researcher` Model: `haiku` / `flash` Workspace: shared read-only	*"Spawn 3 parallel `codebase-researcher` subagents covering `src/auth/`, `src/billing/`, and `src/webhooks/`. Each must return a 250-word report of entry points, external calls, and authz gaps. Merge all three into one prioritized risk table."*
2. Writer + Blind Adversarial Critic Gate Sequential + Bounded Loop	Main builder + `skeptical-reviewer` Cap: 3 rounds max Uses stateful **Resume**	*"Implement cursor pagination on `GET /orders`. Then run `skeptical-reviewer` on the diff. If it returns `FIX`, patch blocking items and **resume the exact same reviewer** to re-verify. Stop at `SHIP` or escalate after 3 rounds."*
3. Competing Hypothesis Bug Hunt Parallel Evidence Race	2x read-only debug scouts Model: `sonnet` / `pro` `maxTurns: 20` each	*"Users see intermittent 502s on checkout. Spawn 2 parallel scouts: Scout A investigates connection-pool exhaustion in `src/db/`; Scout B investigates Stripe webhook retries in `src/payments/`. Require `file:line` proof to crown a winner."*
4. Spec-to-Slice Parallel Feature Build Worktree Builder Fan-Out	2x builders w/ `isolation: worktree` (or `workspace: "share"`) + 1x verifier	*"From our approved API contract, spawn Builder A in a worktree owning `src/api/export/` and Builder B in a worktree owning `src/ui/export/`. Neither may touch outside files. Merge both branches and run `test-writer` end-to-end."*
5. Zero-Regression Migration Pipeline 4-Hop Handoff Chain	`researcher` → Migrator → `test-writer` → `skeptical-reviewer`	*"Run a 4-stage pipeline to upgrade our auth middleware: (1) `codebase-researcher` lists every callsite; (2) migrate callsites in main thread; (3) `test-writer` adds regression coverage; (4) `skeptical-reviewer` verifies backward compatibility."*

The 4-Sentence Briefing Formula

Whenever you (or your coordinator) brief a clean-slate subagent, pack four crisp elements: **1. Goal** (one sentence) • **2. Context** (exact paths + prior stage summary, including disproved dead ends!) • **3. Constraints** (off-limits folders) • **4. Deliverable** (exact report headings).

Context-Repair Rescue Trick

Did a messy search or giant log paste bloat your main conversation? Spawn one fresh subagent to *redo* that investigation cleanly in isolation, keep its 200-word summary, and run `/compact`—buying back sharp model attention immediately!

One Mental Model Across Every Major Coding Agent

In 2026, subagent concepts converged across the entire industry—and several editors even auto-load `.claude/agents/` directly! Use this Rosetta Stone to translate any orchestration workflow in seconds.

Capability	Claude Code	Google Antigravity	Cursor (3.x)	Copilot & OpenAI Codex
Definition Location	`.claude/agents/*.md` `~/.claude/agents/`	Built-in + Plugin agents + dynamic `define_subagent`	`.cursor/agents/*.md` (also reads `.claude/` !)	Copilot: `.github/agents/` Codex: `.codex/agents/*.toml`
Explicit Dispatch	`@agent-name`, prose, or `claude --agent`	`invoke_subagent` tool, `/browser`, `/teamwork`	`/name` slash command in chat or prose	Copilot: `agents:` allowlist Codex: `spawn_agent` prompt
Tool & Sandbox Lock	`tools` / `disallowedTools` + `PreToolUse` hooks	`EnableWriteTools`, `EnableMCPTools`, sandbox	`readonly: true` flag in YAML frontmatter	Copilot: `['read','search']` Codex: `sandbox_mode`
Parallel Git Isolation	`isolation: worktree` in frontmatter	`Workspace: "share"` or `"branch"` clone	Worktree env on request or `/in-cloud` VM	Copilot: cloud PR branch Codex: session worktrees
Context Fork & Resume	`/subtask` fork + `SendMessage` to ID	`fork:"coordinator"` + `send_message(id)`	`Resume agent <id>`; `/goal` long runs	Copilot: UI `handoffs` Codex: `resume_agent`
Peer Team Mode	Agent Teams & cross-session messaging	Native `swarm` board (`/teamwork`)	Agents Window multi-VM & `/autopilot`	VS Code Agent Host (5-deep) & Agent Merge

The Polyglot Repository Layout: Commit Once, Run in Every Client

```
my-repository/
├─ AGENTS.md              # Universal repo rules & cmds
├─ .claude/agents/        # Canonical Markdown agents
│  ├─ researcher.md       # Read by Claude/Cursor/VSCode
│  └─ reviewer.md         # Adversarial diff gate
├─ .agents/skills/        # On-demand SKILL.md playbooks
├─ .cursor/agents/        # Optional Cursor overrides
├─ .github/agents/        # Optional Copilot handoffs
└─ .codex/agents/*.toml   # Optional Codex TOML roles
```

Cross-Compatibility Rules of Thumb

- **Start with `AGENTS.md` + `.claude/agents/`:** Both Cursor and VS Code detect `.claude/agents/*.md` natively, while Antigravity, Codex, and Cursor all obey root `AGENTS.md` instructions.
- **Know what silently drops:** Advanced Claude fields (`hooks`, `permissionMode`, `memory`, `isolation`) are ignored by other clients—keep core role rules in the Markdown body too!

End-to-End Orchestration Pipeline & Cost Engineering

Smart orchestration pairs a **frontier reasoning coordinator** with **fast, inexpensive specialist workers**—delivering cleaner architecture and independent verification at a fraction of an all-frontier swarm bill.



6 Token & Cost Guardrails That Prevent Multi-Agent Bill Shock

1 Tier Models by Judgment Needed

Keep `opus` / `gemini-pro` on your coordinator, pin `haiku` / `flash` on file scouts, and use `sonnet` on builders. Set `CLAUDE_CODE_SUBAGENT_MODEL` as a safe floor.

2 Cap Fan-Out Width at 3–5 Workers

Five parallel subagents spend ~5x tokens, and ten verbose return summaries will overflow your coordinator's context window! Never fan out what one `Grep` can answer.

3 Guard the 15K Description Budget

Every installed agent's `description` loads into session preamble on turn 1. Keep descriptions to 1–2 crisp sentences; put detailed instructions in the Markdown body.

4 Set `maxTurns` & Round Caps

Put `maxTurns: 20–30` on exploratory or review agents, and cap critic loops at 3 rounds where only *blocking* defects trigger another iteration.

5 Use Forks & `cacheTtl: 1h`

Use `/subtask` forks when side tasks need heavy shared history (free prompt-cache hit!), and set `experimental: { cacheTtl: 1h }` on long-running specialists.

6 Enforce Terse Output Contracts

Every token a subagent returns gets billed again on every subsequent turn of your main session. Demand 250-word bulleted memos citing `file:line`.

Troubleshooting Clinic & Pre-Flight Cheat Sheet

Keep this back-page clinic within reach whenever a custom subagent fails to trigger, loses context across stages, or stalls silently in the background.

Top 7 Subagent & Orchestration Pitfalls (& Their 10-Second Fixes)

Symptom / Failure Mode	Root Cause & 10-Second Fix
Coordinator never auto-delegates to my custom agent	Vague `description` (e.g., *"Helps with tests"*). Rewrite as a sharp trigger: *"Use proactively right after code edits to write and run unit tests"*—or pin via `@agent-name`.
Newly created agent file is completely invisible	`.claude/agents/` folder did not exist at session start, or YAML frontmatter has syntax errors / colons in `name`. Restart session and run `claude plugin validate .claude/agents`.
Stage 2 worker repeats Stage 1's mistakes or searches	Amnesia across clean-slate hops. Require Stage 1's output contract to list *disproved dead ends*, pass its report into Stage 2's prompt, or use `/subtask` (fork) / `SendMessage` resume.
Two parallel builders overwrote each other's edits	Shared git checkout with overlapping files. Add `isolation: worktree` (or Antigravity `Workspace: "share"`) per builder, or partition strict non-overlapping directory ownership.
Critic review loop burns tokens forever on tiny nits	Fresh reviewer respawned each round without convergence rules. Always **resume the same reviewer ID**, split *Blocking* vs. *Non-Blocking*, and cap at 3 rounds max.
Background subagent hangs mid-run without finishing	Waiting on an interactive tool approval prompt surfaced in your main window—or folder trust wasn't accepted for project hooks. Answer the prompt or use `acceptEdits` / `dontAsk`.
Built-in `Explore` scout ignored my `CLAUDE.md` rules	By design, lightweight built-in Explore/Plan scouts skip `CLAUDE.md` to save startup tokens. Put critical constraints (like *"ignore `vendor/`"*) directly in the delegation prompt—or use your own custom scout!

Orchestration Pre-Flight Checklist: 6 Habits Before You Dispatch a Fleet

- ☐ **1. Commit the Core Triad First:** `researcher`, `test-writer`, and `skeptical-reviewer`.
- ☐ **2. Enforce Least Privilege in YAML:** Strip `Write` / `Edit` from scouts & critics; gate `Bash` with hooks.
- ☐ **3. Write Output Contracts:** Specify exact headings & `file:line` citations so reports carry cleanly.
- ☐ **4. Isolate Parallel Editors:** Enable `isolation: worktree` or disjoint folder boundaries.
- ☐ **5. Resume By Agent ID:** Continue multi-round reviews via `SendMessage` instead of respawning.
- ☐ **6. Bound Every Loop:** Set `maxTurns` + a 3-round escalation rule before walking away!

Your Fastest 5-Minute Win Today

Drop `codebase-researcher.md` and `skeptical-reviewer.md` (from Page 6) into your repo's `.claude/agents/` folder, commit them to Git, and run your very next feature as **Scout → Build → Blind Critic**. You will feel the jump in context clarity on turn one!