

Orchestrator Context Management in Claude Code

Gauntlet AI | Claude Code Mastery Series

The Question

A common frustration when building multi-agent pipelines is getting the orchestrator to clear its context more often and more automatically. It feels like the session accumulates noise over time, the agent starts to drift, and the instinct is to find a way to trigger `/clear` programmatically.

This guide explains why that instinct, while correct in spirit, is pointing at the wrong solution, and what to do instead.

The Core Constraint: `/clear` is a Client-Side Command

The first thing to understand is that `claude /clear` is an **interactive, client-side command**. It resets the conversation window in the UI. There is no native hook, API event, or agent instruction that can fire it automatically mid-session. Any architecture that depends on auto-clearing the context will eventually break or require a manual step regardless.

The real goal, therefore, is not to trigger `/clear` more often. It is to **design the orchestrator so its context stays lean enough that clearing is rarely urgent in the first place**. When an orchestrator's context is bloating, it is almost always a signal that the orchestrator is doing work it should be delegating.

Solution 1: Sub-Agents as Natural Context Boundaries

This is the highest-leverage fix. Every time the orchestrator spawns a sub-agent (a coder, reviewer, or tester), that agent runs in its **own isolated context window**. It starts fresh, does its job, and returns a compact result summary. The orchestrator's main session only ever sees that summary, not the full working context of the sub-agent.

If the orchestrator's context is growing rapidly, the most common root cause is that heavy work is happening in the main session rather than being pushed into sub-agents. The fix is to delegate more aggressively, not to clear more aggressively.

The mental model to internalize: the orchestrator should be a thin coordinator. Its job is to classify, plan, delegate, and track. The moment it starts reading large files, writing code, or running commands itself, context bloat is inevitable. Every unit of work that moves into a sub-agent is a unit of context that never touches the main session.

Solution 2: Deterministic /clear Checkpoints in ORCHESTRATION.md

For the cases where the main session context genuinely does accumulate over a long working session, the correct approach is to define **explicit, rule-based checkpoints** in ORCHESTRATION.md where the agent is instructed to prompt the user to run /clear. This makes the reminder automatic and tied to measurable events rather than a vague sense that things feel slow.

Add a section like the following to ORCHESTRATION.md:

```
## Context Reset Checkpoints

After each of the following events, output this message to the user:
"Context checkpoint reached. Run /clear to reset the session
before continuing. All epic and story state is saved in epics.json."

Trigger points:

After a story is fully merged into the epic branch
After the epic PR is opened or updated
After 3 or more stories have been completed in a single session
After any session recovery from epics.json
```

This approach keeps the actual /clear as a deliberate user action (which is correct, since it is destructive) while making the prompt to do so automatic and consistent.

Solution 3: A PostToolUse Hook for Lightweight Reminders

For a more automated implementation, a PostToolUse hook can fire after key task state changes and inject a context reminder. This does not replace the checkpoints above, but it keeps the reminder visible in the agent's context at the right moments without requiring any external state or complex logic.

Add the following to .claude/settings.json:

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "TaskUpdate",
        "hooks": [
          {
            "type": "command",
            "command": "echo 'Reminder: if 3 or more stories
are complete this session, prompt the user to
run /clear before starting the next story.'"
          }
        ]
      }
    ]
  }
}
```

The hook fires after every `TaskUpdate` call and echoes the reminder back into context. It is lightweight, requires no dependencies, and degrades gracefully if the threshold has not been reached yet.

Putting It Together

These three solutions work at different layers and complement each other:

Layer	Mechanism	What It Solves
Sub-agent delegation	Push all heavy work into sub-agents	Prevents context from bloating in the first place
ORCHESTRATION.md checkpoints	Rule-based prompts tied to story and epic events	Ensures the user is reminded to /clear at the right moments
PostToolUse hook	Lightweight reminder injected after task state changes	Keeps the reminder visible automatically

The key shift in thinking is this: **if you find yourself wanting to clear context more often, treat that as a diagnostic signal, not just an inconvenience.** It almost always means the orchestrator is doing too much work itself. Fixing the delegation is a more durable solution than any amount of context clearing.