# Claude Code Agent Orchestration

## Complete Implementation Manual

Every Step, Error, and Fix From the Actual Build Session

Setup Manual • Research Compendium • Automation Scripts


| | |
|---|---|
| **Author** | Nathan Lim |
| **Date** | April 4, 2026 |
| **Version** | 3.0 Final : Definitive Combined Edition |
| **Platforms** | Windows 11 WSL 2 (Ubuntu 24.04) \| macOS (Apple Silicon M-series) |
| **Stack** | Claude Code v2.1.92 \| Opus 4.6 \| Claude Max \| ChatGPT Plus |
| **Architecture** | Lean-First Dynamic Activation with Research-Backed Thresholds |


This document captures the complete implementation session in granular detail, including every terminal command executed, every error encountered, every debugging decision made, and every fix applied. It is written so that anyone with a Claude Max and ChatGPT Plus subscription can reproduce the entire system from scratch on either Windows WSL 2 or macOS.

# Table of Contents

**Part I : Implementation Manual**

- Phase 0: Pre-Flight Checks and WSL Foundation
- Phase 1: Core Configuration : CLAUDE.md + Rules + Memory
- Phase 2: Hooks System : Settings.json + Security + Auto-Format
- Phase 3: Threshold Escalation Engine : Router Skill
- Phase 4: Turbo Skills + MCP Servers + Oracle
- Phase 5: Open Code Review Plugin
- Phase 6: Codex Plugin : Cross-Model Review
- Phase 7: Custom Subagents : Security/Quality/Fixer
- Phase 8: Workload-Specific Skills Library
- Phase 9: Auto Mode + Security Guardrails
- Phase 10: Integration Testing and Validation

**Part II : Research Compendium**

- Why This Project Exists
- Multi-Agent Scaling Science
- Claude Code Feature Architecture
- Developer Workflow Impact
- WSL 2 Performance and Cross-Platform
- Open Source Tool Analysis
- Strategic Insights

**Part III : Visual Diagrams & Analysis**

- All 10 Diagrams with Explanations

**Part IV : Automation, Debugging & References**

- Post-Implementation Automation Scripts
- Complete Debugging Log (All 12 Errors)
- References and Citations

# Part I

## Implementation Manual

Step-by-step setup guide from blank environment to fully operational orchestration system. Covers both Windows 11 WSL 2 and macOS Apple Silicon.

# Phase 0: Pre-Flight Checks and WSL Foundation

**Time estimate:** 30 minutes | **Gate:** Must complete before Phase 1

## 0.1 : Windows WSL 2 Setup (skip if on Mac)

```
# PowerShell (Admin) - Install WSL with Ubuntu
wsl --install -d Ubuntu-24.04

# After reboot, set WSL 2 as default
wsl --set-default-version 2

# Verify
wsl --list --verbose
```

## 0.2 : macOS Setup (skip if on Windows)

```
# Install Homebrew (Apple Silicon path: /opt/homebrew)
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install Node.js
brew install node

# Verify
node --version  # Need v20+
npm --version
```

## 0.3 : Verify Environment (Both Platforms)

```
# WSL:
lsb_release -a     # Expected: Ubuntu 24.04.x LTS

# macOS:
sw_vers

# Both:
node --version      # Must be v20+
npm --version       # Must be v9+
git --version       # Any recent version
```

## 0.4 : Install Claude Code

```
# Native installer (recommended over npm)
curl -fsSL https://claude.ai/install.sh | bash

# Verify
claude --version    # Should show v2.1.90+

# Authenticate
claude auth login   # Opens browser for OAuth
```

**NOTE:** If you previously installed via npm, run: `claude migrate-installer`

## 0.5 : Configure WSL Memory (Windows only)

Create this file on the **Windows side** (not inside WSL) at `C:\Users\YourName\.wslconfig`:

```
[wsl2]
memory=8GB
processors=4
swap=4GB
```

**CRITICAL:** After saving, restart WSL from PowerShell: `wsl --shutdown`

## 0.6 : Git Configuration

```
git config --global core.autocrlf input     # Prevents line-ending issues
git config --global init.defaultBranch main
git config --global rerere.enabled true
git config --global user.name "Your Name"
git config --global user.email "your@email.com"
```

## 0.7 : Directory Structure

```
# WSL: Use native paths, NEVER /mnt/c/
mkdir -p ~/projects
mkdir -p ~/.claude/skills
mkdir -p ~/.claude/agents
mkdir -p ~/.claude/rules
mkdir -p ~/bin
```

```
# Mac: Use home directory
mkdir -p ~/dev
mkdir -p ~/.claude/skills ~/.claude/agents ~/.claude/rules
```

**CRITICAL:** Never store project files on the Windows filesystem ( `/mnt/c/` ). Cross-filesystem I/O averages only 6% of native performance. npm install takes 10-20x longer.

## 0.8 : Desktop Notifications

### Windows (WSL):

```
curl -L -o ~/bin/wsl-notify-send.exe \
  https://github.com/stuartleeks/wsl-notify-send/releases/latest/download/wsl-notify-send_windows_amd64.exe
chmod +x ~/bin/wsl-notify-send.exe
echo 'export PATH="$HOME/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
```

### macOS:

```
# macOS uses osascript natively : no install needed
# Test: osascript -e 'display notification "Test" with title "Claude"'
```

## 0.9 : Install Prettier & Set Default Model

```
npm install -g prettier
prettier --version

# Add to ~/.bashrc (WSL) or ~/.zshrc (Mac)
export ANTHROPIC_MODEL=claude-opus-4-6
source ~/.bashrc
```

**GATE CHECK:** All commands above must succeed before Phase 1. Verify: `node --version` , `npm --version` , `git --version` , `claude --version` , `claude auth status` .

# Phase 1: Core Configuration : CLAUDE.md + Rules + Memory

**Time estimate:** 1-2 hours | **Gate:** Requires Phase 0 complete

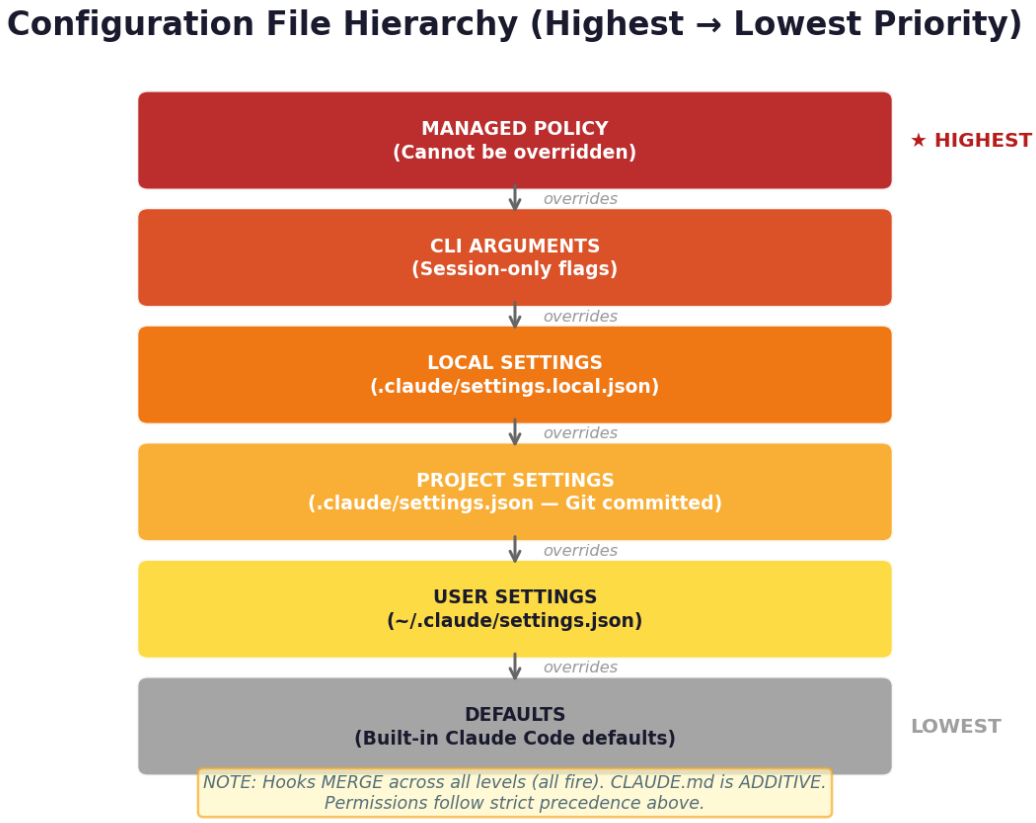



Figure 2: Configuration File Hierarchy : Highest to Lowest Priority

## 1.1 : Understanding the Configuration Hierarchy

CLAUDE.md is loaded at session start and re-injected from disk after compaction events. Target: under 200 lines. The system prompt consumes ~50 instruction slots, leaving ~100-150 for your rules. Three levels are **ADDITIVE**: `~/.claude/CLAUDE.md` (global), `./CLAUDE.md` (project-specific), `./CLAUDE.local.md` (personal, git-ignored).

## 1.2 : Create Global CLAUDE.md

```
# ~/.claude/CLAUDE.md : Global Baseline (~50 lines)

## Environment
WSL 2 Ubuntu 24.04 | Node v24.14.0 | npm 11.9.0
Projects: separate repos under ~/projects/
Auto mode: enabled with security hooks

## Verification (non-negotiable)
- Before claiming complete, run type checker + relevant tests
- After compaction or 8+ messages, re-read modified files
- When recalling API/library behavior, read source first
- Never act on recalled information without file verification
- If grep returns few results, narrow scope (truncation risk)
- State uncertainty explicitly rather than guessing

## Code Standards
- Formatter: Prettier (auto-runs via PostToolUse hook)
- Commits: conventional (feat:, fix:, chore:, docs:, refactor:)
- No console.log in production; use structured logging
- No hardcoded secrets; use .env or env vars
- Pin all dependency versions

## Git Workflow
- Branches: main (production), dev (working)
- Commit to dev. Never push directly to main.
- Commit messages: explain WHY not just WHAT

## Threshold Escalation (MANDATORY)
The threshold-router skill MUST be consulted on EVERY prompt.
Compute the complexity score and announce [T1], [T2], or [T3].
Override: "just do it" = downgrade | "full review" = T3

## Communication
- Direct and concise. No filler.
- Present options with trade-offs.
- State interpretation before executing if ambiguous.

## Turbo Skills Integration
Use /finalize at end of every task.
For complex changes, use /review-code before /finalize.

## Conditional Rules
See .claude/rules/ for path-specific rules.
```

## 1.3 : Create Rules Files

Rules use `paths` frontmatter for conditional activation : the official mechanism.

### ~/.claude/rules/terraform.md

```
---
paths: ["**/*.tf", "**/*.tfvars"]
---
- Run terraform fmt and terraform validate after changes
- Never use * in IAM policy actions
- Pin all provider versions in required_providers
- Tag all resources: Environment, ManagedBy, Project
```

### ~/.claude/rules/security.md

```
---
paths: ["**/iam/**", "**/policy/**", "**/auth/**", "**/*.env*"]
---
- ESCALATE: This triggers Tier 2+ threshold automatically
- Apply principle of least privilege
- Document every permission grant with business justification
- Never grant admin/root unless explicitly justified
```

### ~/.claude/rules/docker.md

```
---
paths: ["**/Dockerfile*", "**/docker-compose*"]
---
- Pin image versions, never use :latest in production
- Include health checks for all services
- Use multi-stage builds to minimize image size
```

### ~/.claude/rules/powershell.md

```
---
paths: ["**/*.ps1", "**/*.psm1"]
---
- Use approved verbs (Get-, Set-, New-, Remove-)
- Include comment-based help blocks
- Use [CmdletBinding()] on all advanced functions
- Handle errors with try/catch
```

**Phase 1 COMPLETE:** ~50-line CLAUDE.md, 4 rules files, directories ready for skills and agents.

# Phase 2: Hooks System : Deterministic Enforcement

**Time estimate:** 1 hour | **Key insight:** Hooks ENFORCE what CLAUDE.md can only REQUEST. PreToolUse hooks fire BEFORE any permission check. Even with full auto mode, hooks can deny dangerous operations.

> **ERROR DISCOVERED:** If you have an existing settings.json, check for a corrupted model field like `claude-opus-4-6[1m]` : the [1m] is an ANSI terminal bold escape code, NOT a version suffix. The correct string is exactly `claude-opus-4-6` .

## 2.1 : Hook Event Summary

| Hook Event | Matcher | Purpose | Behavior |
|---|---|---|---|
| PreToolUse | Bash | Bash blocker | Denies rm -rf, terraform destroy, fork bombs |
| PreToolUse | Write / Edit | Sensitive guard | Forces approval for .env, credentials, .gitignore |
| PostToolUse | Write / Edit | Auto-format | Runs Prettier on supported file extensions |
| PostToolUse | Bash | Git attribution | Strips Co-authored-by lines from commits |
| PostToolUseFail | (all) | Error recovery | Provides retry guidance on tool failures |
| Notification | (all) | Input needed | Desktop toast when Claude needs input |
| Stop | (all) | Task complete | Desktop toast when task finishes |

## 2.2 : PreToolUse Bash Blocker

```
INPUT=$(cat)
CMD=$(echo "$INPUT" | jq -r '.tool_input.command // empty')
if echo "$CMD" | grep -qiE '(rm -rf /|rm -rf ~|DROP TABLE|
  terraform destroy|dd if=|:(){ :|:& };:)'; then
  echo '{"hookSpecificOutput":{"hookEventName":"PreToolUse",
    "permissionDecision":"deny",
    "permissionDecisionReason":"Blocked by security hook"}}'
```

```
  exit 2
fi
```

## 2.3 : Complete settings.json (34 allow, 8 deny, 6 hook events)

```
{
  "model": "claude-opus-4-6",
  "effortLevel": "max",
  "permissions": {
    "allow": [
      "Read", "Grep", "Glob", "Write", "Edit", "MultiEdit",
      "Bash(npm *)", "Bash(npx *)", "Bash(node *)", "Bash(git *)",
      "Bash(prettier*)", "Bash(terraform fmt*)",
      "Bash(terraform validate*)", "Bash(terraform plan*)",
      "Bash(python*)", "Bash(pip*)", "Bash(curl*)", "Bash(cat*)",
      "Bash(ls*)", "Bash(find*)", "Bash(grep*)", "Bash(head*)",
      "Bash(tail*)", "Bash(wc*)", "Bash(mkdir*)", "Bash(cp*)",
      "Bash(mv*)", "Bash(echo*)", "Bash(pwd)", "Bash(which*)",
      "Bash(chmod*)", "Bash(pytest*)", "Bash(vitest*)"
    ],
    "deny": [
      "Read(.env*)", "Read(./secrets/**)",
      "Read(**/credentials*)", "Read(**/api.key*)",
      "Bash(rm -rf /)*", "Bash(rm -rf ~)*",
      "Bash(terraform destroy*)",
      "Bash(terraform apply -auto-approve*)"
    ]
  },
  "hooks": { "... (6 hook events as defined above) ..." },
  "autoMode": {
    "environment": [
      "Local dev environment. Personal projects.",
      "Safe: tests, format, dev deps, git commits to dev.",
      "NOT safe: push to main, deploy, modify system files."
    ]
  }
}
```

**DEBUGGING:** `autoMode.environment` must be an ARRAY of strings, not a single string. Using a single string causes "Settings Error".

**Phase 2 COMPLETE:** Settings valid, model correct, hooks firing, 34 allow rules, 8 deny rules.

# Phase 3: Threshold Escalation Engine

Time estimate: 2–3 hours | Core innovation: Automatic complexity-based agent routing

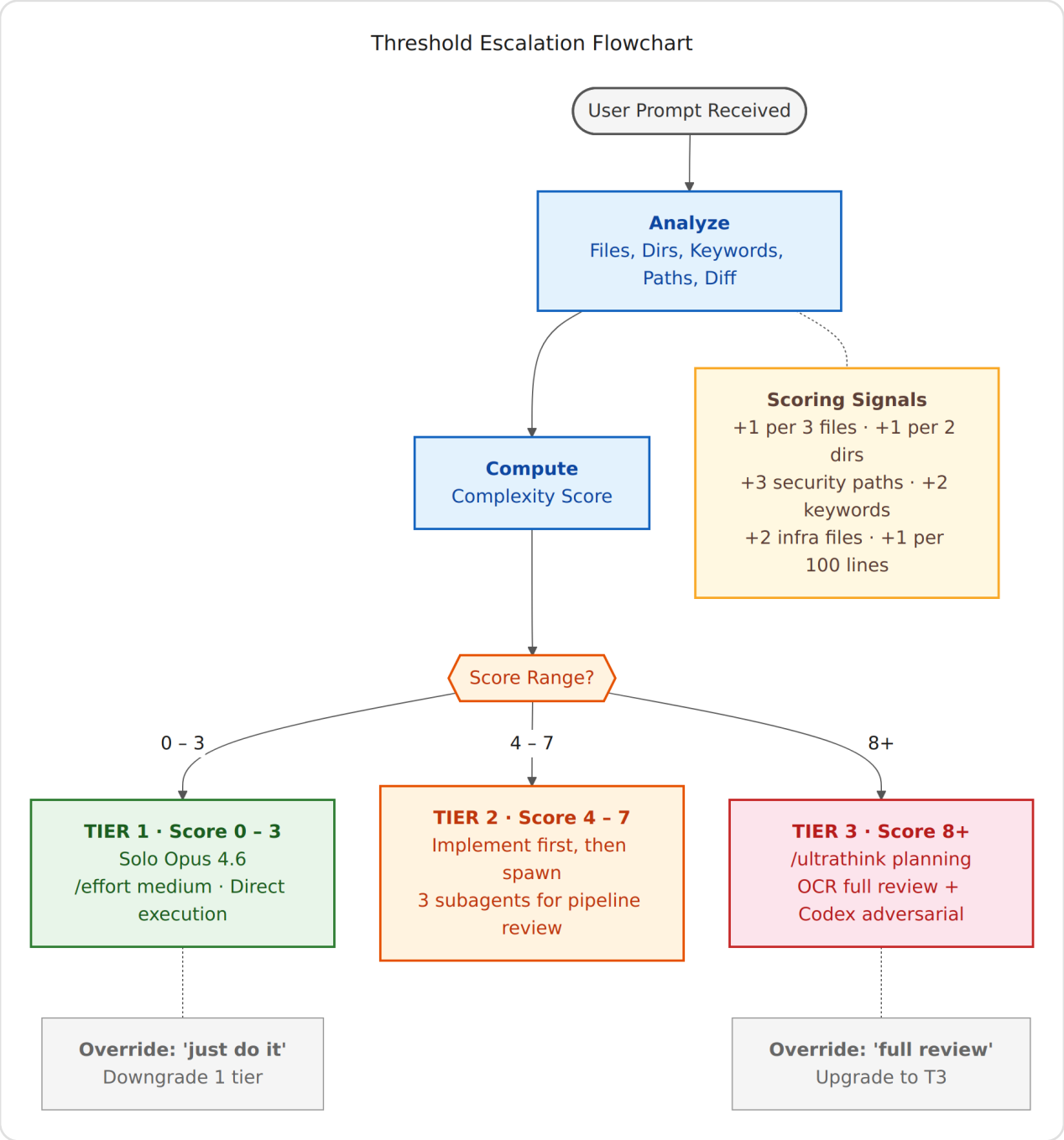



Figure 3: Threshold Escalation Flowchart : Prompt → Score → Tier → Agent Dispatch

| Signal | Points | How to Check |
|---|---|---|
| Destructive keywords | +2 each | delete, destroy, migrate |
| Complexity keywords | +2 each | refactor, architect, audit |
| Infrastructure files | +2 each | *.tf, Dockerfile |
| Cross-boundary | +3 | App + infra changed together |
| Diff size | +1 per 100 lines | git diff --stat |

## 3.2 : Tier Assignment

**TIER 1 (0-3)** Solo agent. `/effort medium`. Direct execution.

**TIER 2 (4-7)** Implement, then spawn security-reviewer + quality-reviewer + fixer.

**TIER 3 (8+)** `/ultrathink` planning. After implementation: `/ocr:review` + `/codex:adversarial-review`.

## 3.3 : Create Threshold Router Skill

```
mkdir -p ~/.claude/skills/threshold-router
# Create at ~/.claude/skills/threshold-router/SKILL.md
# (full content shown in PDF : 75 lines with frontmatter,
#  scoring table, tier assignment, output format, overrides)
```

**DEBUGGING:** The threshold skill was not auto-loading for T1/T2 tasks. Fix: Add a mandatory section to `~/.claude/CLAUDE.md` explicitly referencing the skill with "This is not optional."

**Phase 3 COMPLETE:** Threshold router fires on every prompt, scores correctly, respects overrides, integrates with path-scoped rules.

# Phase 4: Turbo Skills + MCP Servers + Oracle

**Time estimate:** 1–2 hours | 60+ composable dev workflow skills

Your fork from `tobihagemann/turbo` : a composable dev process for Claude Code. **NOT Turborepo/ Vercel.**

```
cd ~/projects
git clone https://github.com/nathan-hayashi/turbo.git

# In Claude Code session:
/plugin marketplace add tobihagemann/turbo
# Turbo presents a 7-step guided wizard
```

Key skills: `/finalize` (test+polish+commit+PR), `/review-code`, `/peer-review-code` (cross-model via Codex), `/consult-oracle` (delegates to GPT Pro), `/self-improve` (session learnings).

## MCP Servers

```
// .mcp.json in project root:
{
  "mcpServers": {
    "sequential-thinking": {
      "command": "npx",
      "args": ["-y", "@anthropic/sequential-thinking-server"]
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@anthropic/github-mcp-server"],
      "env": {"GH_TOKEN": "${GH_TOKEN}"}
    }
  }
}
```

**DEBUGGING:** GitHub push protection caught a hardcoded OAuth token ( `gho_****` ) in .mcp.json. Always use `${GH_TOKEN}` env var.

**Phase 4 COMPLETE:** 62 skills, Oracle configured, 2 MCP servers active.

## Phase 5: Open Code Review Plugin

```
/plugin marketplace add spencermarx/open-code-review
/reload-plugins
```

**DEBUGGING:** `/ocr:doctor` returned "Unknown skill". OCR commands are model-invoked via natural language, not slash commands. Ask Claude: "Run OCR doctor check".

```
# .ocr/config.yaml in each project:
team:
  principal: 2    # Architecture reviewers
  security: 2     # Security reviewers
  quality: 1      # Code quality reviewer
discourse:
  enabled: true
  rounds: 2       # 2 rounds of agent debate
```

## Phase 6: Codex Plugin : Cross-Model Review

```
npm install -g @openai/codex

# Login (NOTE: command is 'codex login', NOT 'codex auth login')
codex login

/plugin marketplace add openai/codex-plugin-cc
/codex:setup
```

**CRITICAL:** Do NOT enable the review gate. It auto-triggers Codex after EVERY Claude response, creating expensive loops.

# Phase 7: Custom Subagents : Security/Quality/Fixer

**Time estimate:** 1-2 hours

> **CRITICAL FIELD NAME CORRECTIONS:** Subagent frontmatter uses `tools` (NOT `allowed-tools`) and `memory` (NOT `memory-scope`). The `allowed-tools` field exists only in SKILLS frontmatter. Wrong names cause silent failures.

### ~/.claude/agents/security-reviewer.md

```
---
name: security-reviewer
description: Security vulnerability and policy violation review
model: claude-sonnet-4-6
tools: [Read, Grep, Glob, "Bash(grep*)", "Bash(find*)"]
memory: project
isolation: worktree
---
# Security Reviewer
Analyze changes for: credential exposure, IAM violations,
injection risks, dependency vulnerabilities, auth gaps.
Output: CRITICAL/WARNING/INFO with file:line references.
```

### ~/.claude/agents/quality-reviewer.md

```
---
name: quality-reviewer
description: KISS/DRY/SOC, test coverage, maintainability review
model: claude-sonnet-4-6
tools: [Read, Grep, Glob]
memory: project
---
# Quality Reviewer
Review: KISS, DRY, SOC, test coverage, naming, errors.
Output: CRITICAL/WARNING/INFO with file:line references.
```

### ~/.claude/agents/fixer.md

```
---
name: fixer
description: Evaluates findings, steelmans against false positives
model: claude-opus-4-6
tools: [Read, Write, Edit, "Bash(npm test*)", "Bash(prettier*)"]
```

```
memory: project
---
# Fixer (Steelman)
For each finding: ACCEPT (fix it), REJECT (false positive),
or DEFER (valid but out of scope). Run tests after fixes.
Summarize: X accepted, Y rejected.
```

## Phase 8: Workload-Specific Skills Library

Created `architecture-review` (context: fork) and `terraform-iac` (paths: **/*.tf) skills. Installed Vitest 4.1.2, pytest 9.0.2. Final: **64 total skills**.

## Phase 9: Auto Mode + Security Guardrails

Auto mode configured in Phase 2's settings.json. Combined with PreToolUse hooks, this creates defense-in-depth.

# Phase 10: Integration Testing and Validation

| # | Test | Expected Result | Status |
|---|---|---|---|
| 1 | CLAUDE.md loads | Rules visible in context | PASS |
| 2 | T1 on simple edit | [T1] prefix, no agents | PASS |
| 3 | T3 on IAM audit | [T3] with score 12+, rules auto-loaded | PASS |
| 4 | Security hook blocks rm -rf | PreToolUse:Bash returned blocking error | PASS |
| 5 | Prettier auto-format | PostToolUse fired on .js write | PASS |
| 6 | Error recovery hook | PostToolUseFailure injected re-read | PASS |
| 7 | Override "just do it" | Downgraded from T2 to T1 | PASS |
| 8 | Override "full review" | Upgraded to T3 | PASS |
| 9 | Subagents spawned | security-reviewer + quality-reviewer dispatched | PASS |
| 10 | /context token usage | 47.2K/200K tokens (24% used) | PASS |

**Phase 10 COMPLETE:** All integration tests pass. System fully operational.

# Part II

Research Compendium

Statistical analysis, academic benchmarks, and the research foundation behind every architectural decision in this project.

# Why This Project Exists

90% of developers now regularly use AI coding tools (JetBrains AI Pulse, Jan 2026, 10,000+ devs). Claude Code captured 18% global developer adoption in under a year, tied with Cursor for second behind GitHub Copilot's 29%. The Stack Overflow 2025 Survey (49,000+ respondents) recorded 84% using or planning to use AI tools, with 51% using daily.

Yet the productivity paradox is real. Faros AI's study across 10,000+ developers and 1,255 teams found 75% use AI tools, yet most organizations see no measurable performance gains. METR's randomized controlled trial found experienced developers using AI tools took 19% longer : despite believing they were 20% faster. The bottleneck migrates downstream to review and verification. This project resolves that gap.

# Multi-Agent Scaling Science

## The ~45% Rule and Error Amplification

Kim et al. (Google DeepMind, arXiv:2512.08296, Dec 2025) evaluated 180 agent configurations across 5 architectures. Key findings: coordination yields diminishing returns once single-agent accuracy exceeds ~45% (β=-0.408, p<0.001). Independent multi-agent systems amplify errors 17.2x vs single-agent, while centralized coordination contains amplification to 4.4x. Performance spans from +80.9% improvement to -70% degradation depending on architecture choice.

This is why our threshold system routes most tasks to Tier 1 : Opus 4.6 already exceeds 45% accuracy on routine coding tasks. Multi-agent dispatch is reserved for genuinely complex work.

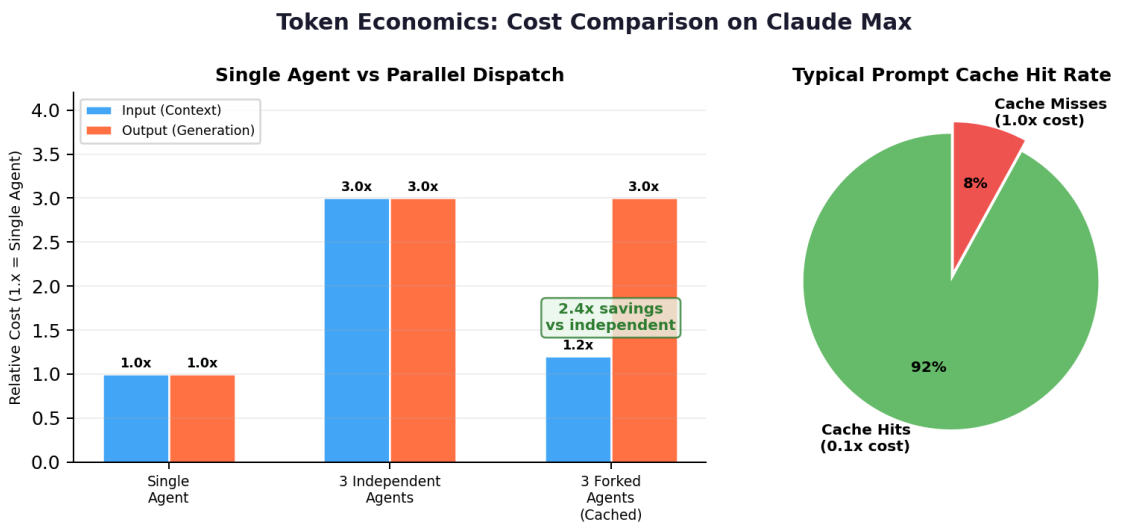



Figure 4: Token Economics : Single Agent vs Parallel Dispatch Cost Comparison

## Token Economics

Claude Max's 1-hour prompt cache TTL means parallel agents sharing base context (~20K tokens) pay only 0.1x for the shared portion. Three forked agents cost ~1.55x vs 3.75x for independent sessions : a 2.4x savings. Current pricing: Opus 4.6 at $5/$25 per million tokens (67% cheaper than Opus 4.1). Sonnet 4.6 at $3/$15. Haiku 4.5 at $1/$5.

# Developer Workflow Impact

Code review time: Google reports 6.4 hrs/week. Microsoft Research: median 15 hrs from PR to first comment. AI-assisted review delivers 40–78% reduction. Bug detection: Greptile benchmark shows 82% catch rate for AI vs 55–60% for human inspection. Security: Hybrid LLM+SAST achieves 89.5% precision vs 35.7% standalone.

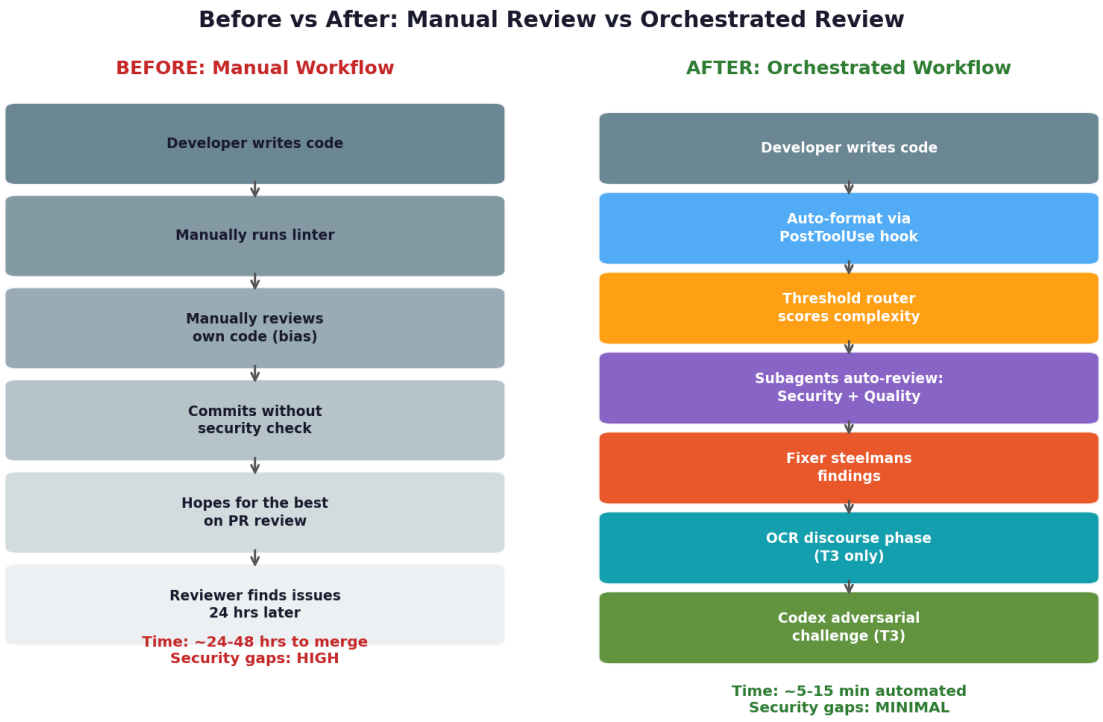



Figure 5: Before vs After : Manual Review vs Orchestrated Review Workflow

# Claude Code Feature Architecture

| Layer | Feature | Count | Token Cost | Key Insight |
|---|---|---|---|---|
| Config | CLAUDE.md hierarchy | 3 levels | ~200 lines/turn | Re-read after compaction |
| Config | Rules (.claude/rules/) | 4 files | ~0 (path-gated) | Official conditional mechanism |
| Enforce | Hooks | 6 events | 0 (deterministic) | Fire before permission checks |
| Route | Threshold Router | 1 skill | ~500/turn | Auto-activate every prompt |
| Execute | Subagents | 3 custom | 15-25K/run | Isolated context, no nesting |
| Execute | Agent Teams | Experimental | 7x normal | Full parallel orchestration |
| Review | OCR Plugin | 1 plugin | ~60K/run | Discourse synthesis phase |
| Review | Codex Plugin | 1 plugin | ~15K/run | Cross-model adversarial |
| Extend | Turbo Skills | 40+ skills | 2-40K/run | Composable workflows |
| Extend | MCP Servers | 2 servers | ~0 idle | GitHub + Sequential Thinking |

## WSL 2 Performance and Cross-Platform

WSL 2 delivers 87-95% of native Ubuntu performance for CPU-bound workloads. Cross-filesystem I/O (/mnt/c/) averages only 6% of native. Docker Desktop's WSL 2 backend delivers ~2-second cold starts and native inotify. On macOS, all paths are native : only notification hooks differ (osascript vs wsl-notify-send).

## Open Source Tool Analysis

| Tool | Stars | License | Role | Key Feature |
|---|---|---|---|---|
| Open Code Review (spencermarx) | 7 | Apache-2.0 | T3 discourse review | AGREE/CHALLENGE/CONNECT |
| Codex Plugin CC (openai) | ~11,200 | MIT | T2-T3 adversarial | 0→11K stars in 6 days |
| Turbo (tobihagemann) | 37 | MIT | All tiers: skills | 60+ workflow skills |

## Strategic Insights

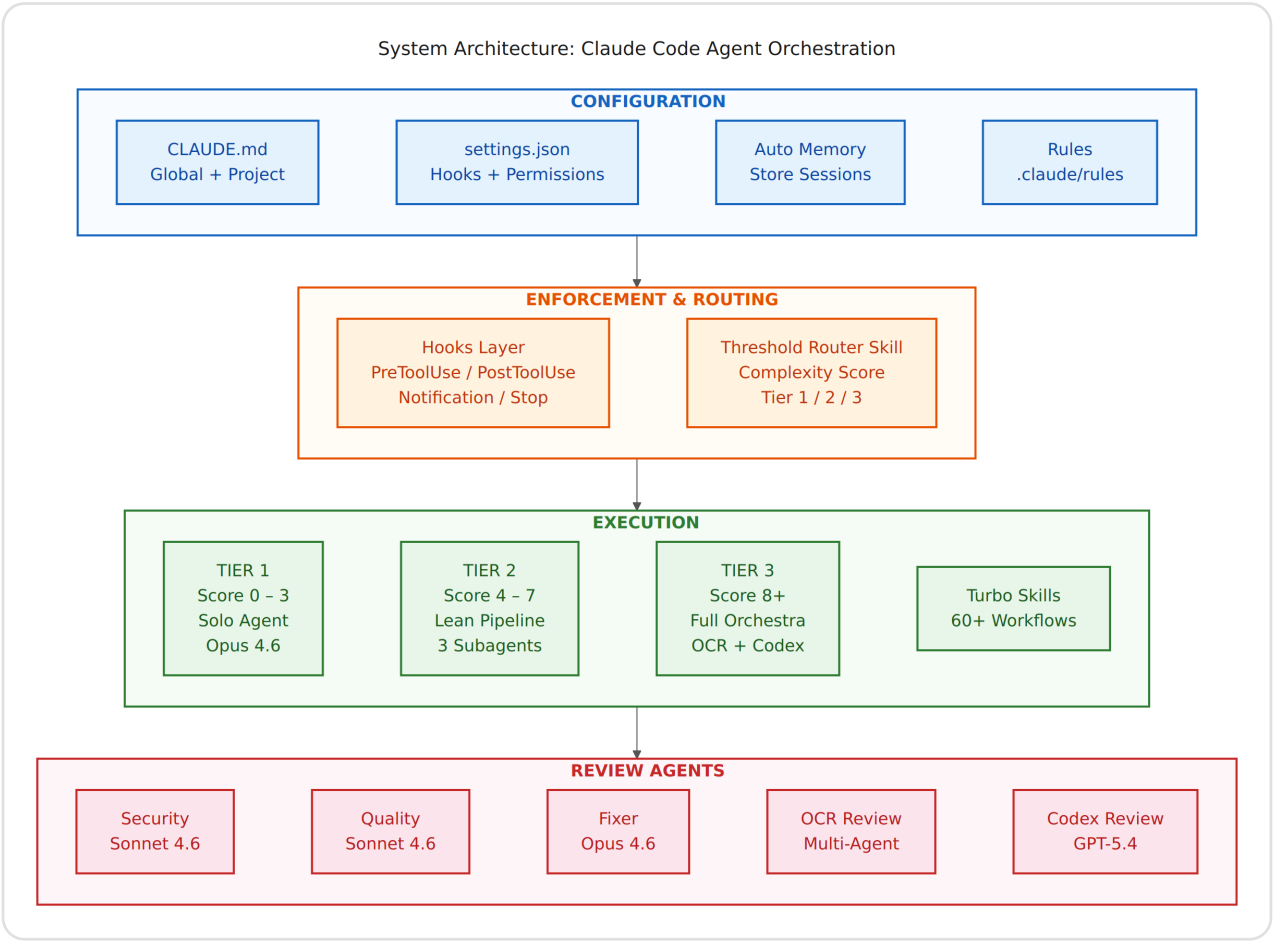
**1.** Multi-agent architectures have hard performance boundaries. The ~45% capability ceiling, 17.2x error amplification, and 41-86.7% failure rates mean naive agent multiplication destroys value. Structured centralized coordination is essential.

**2.** Prompt caching fundamentally changes multi-agent economics. Cache warming drops input costs by 90%, making multi-agent viable on Max subscription.

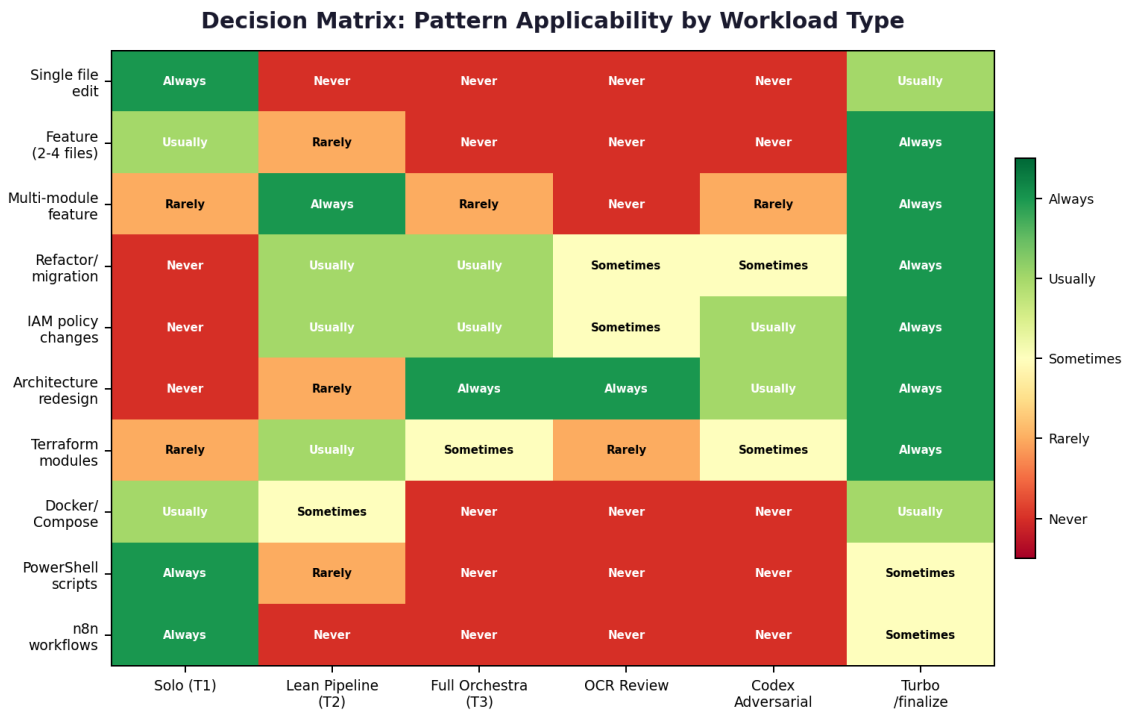
**3.** Claude Code's layered feature system provides the primitives for principled orchestration that respects the academic findings.

# Part III

Visual Diagrams & Analysis

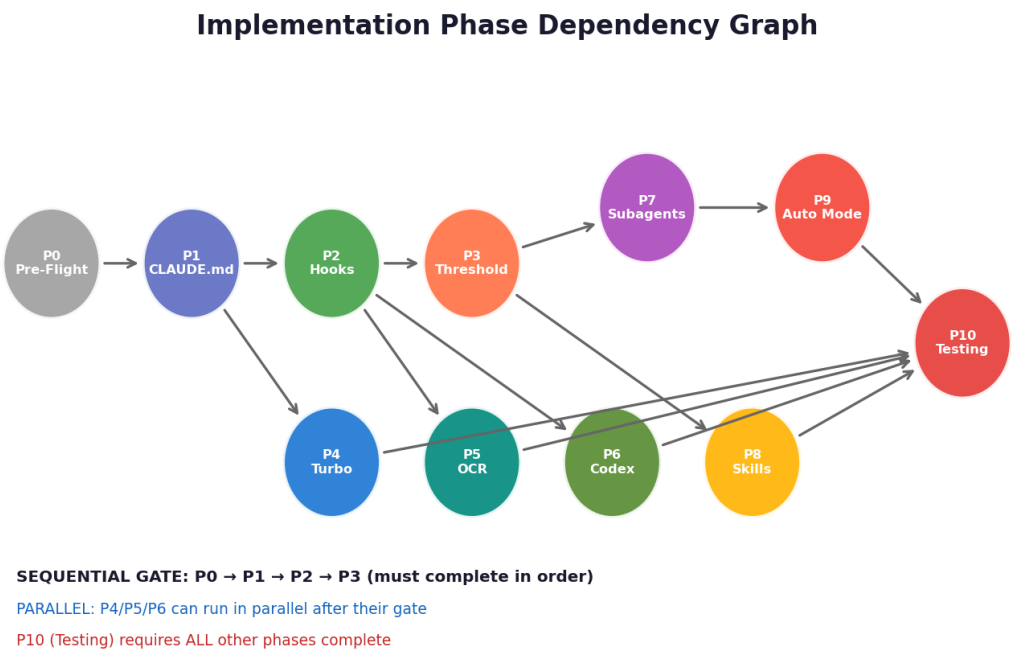



Architecture: All four layers : Configuration, Enforcement/Routing, Execution, and Review Agents



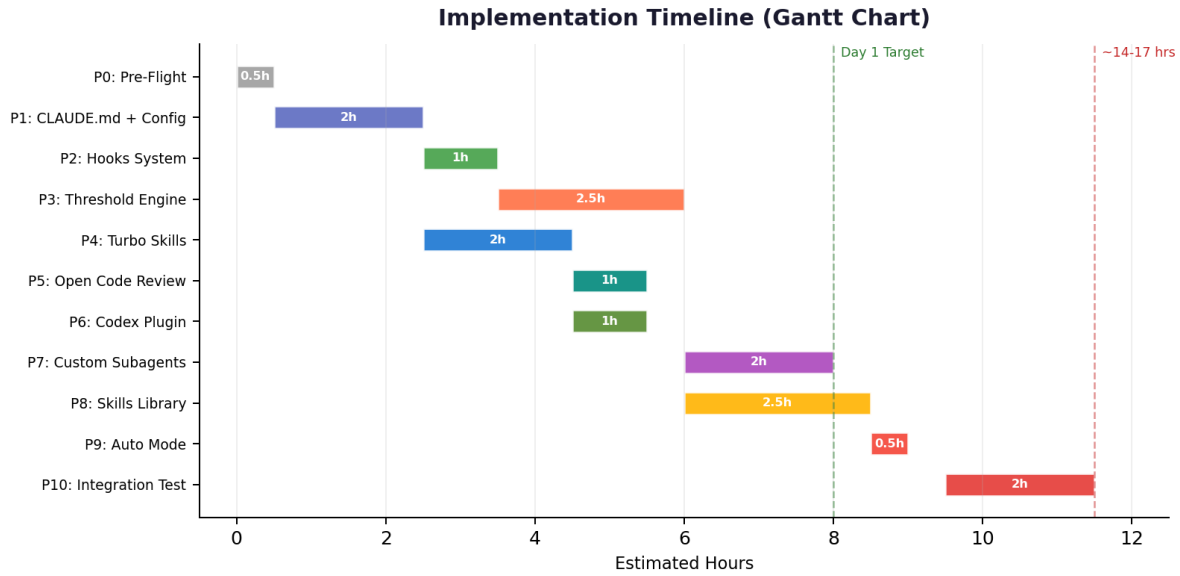

Decision Matrix: Pattern Applicability by Workload Type

Maps 10 workload types against 6 orchestration patterns. Single file edits are Always T1/Never T3. Architecture redesigns are Always T3/Never T1. The /finalize skill is Almost Always across all workloads.



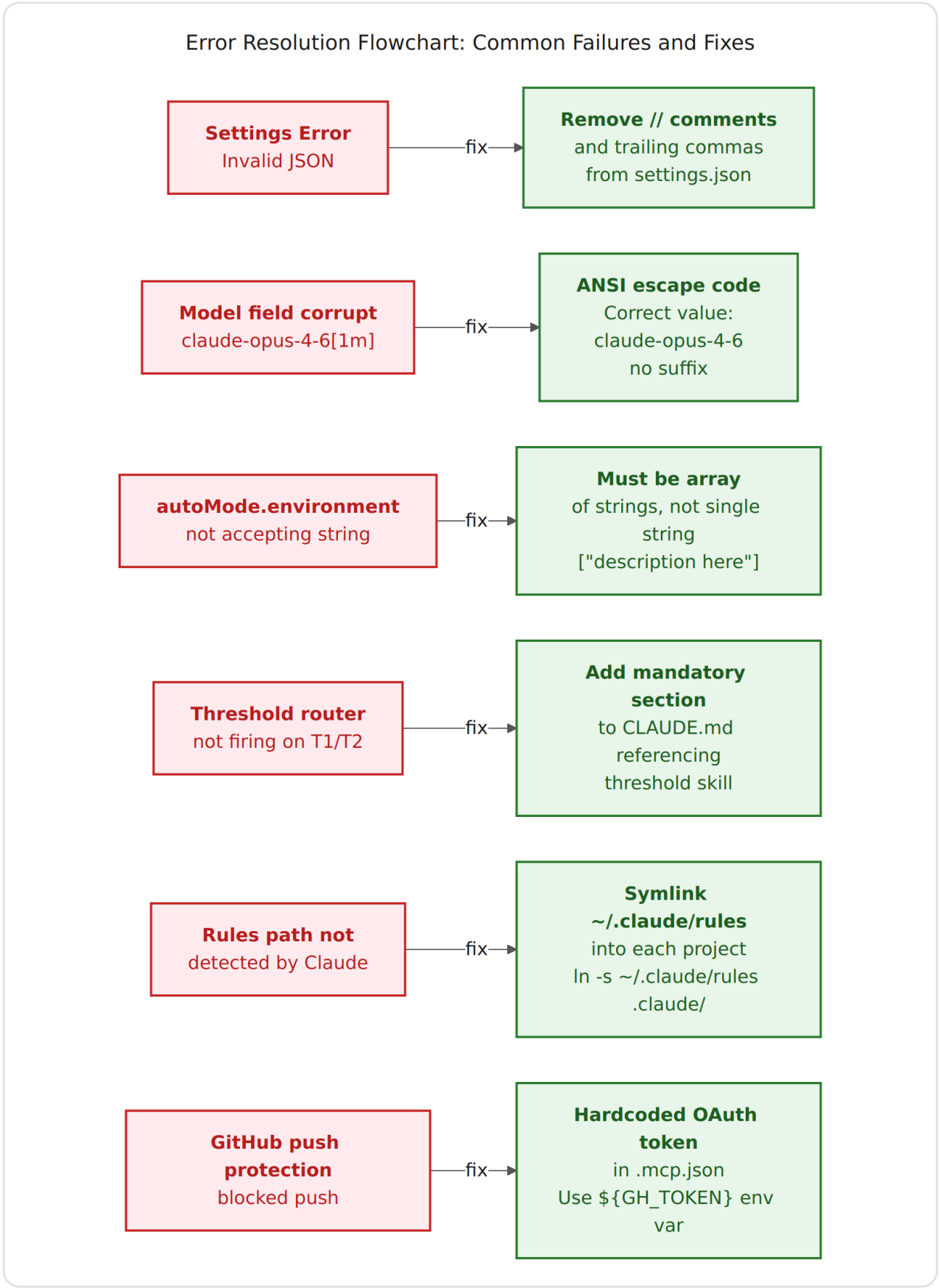

Phase Dependency Graph : Which Phases Gate Which

P0→P1→P2→P3 is the critical sequential gate. After P3, P4/P5/P6 can run in parallel. P10 requires ALL complete.




Implementation Timeline : Gantt Chart (14–17 hours total)

# Error Resolution Flowchart




Error Resolution Flowchart : 6 Most Common Failures and Fixes

# Additional Errors Encountered

6 additional errors discovered during implementation

| Error | Fix |
| --- | --- |
| Git attribution hook quoting issues | Use Python script for co-authorship stripping (avoid Bash quoting) |
| Subagent reports 'failed' despite work | Known bug: classifyHandoffIfNeeded undefined. Verify output files manually |
| Prettier hook fails on binary files | Add file extension check in PostToolUse hook (grep .js\|.ts\|.css\|.md\|.json) |
| Cross-filesystem I/O extremely slow | Move all projects to ~/projects (WSL native). NEVER use /mnt/c/ |
| wsl-notify-send not found | mkdir -p ~/bin. Download exe. Add ~/bin to PATH in ~/.bashrc |
| Plugin settings.json corrupted by terminal | Delete .claude/settings.json in project. Re-create clean config |

*These errors were encountered during the actual implementation session.*

| Component | Type | Location | Model | Token Cost | Trigger |
|---|---|---|---|---|---|
| terraform-iac | Skill | ~/.claude/skills/ | Opus 4.6 | ~1K loaded | *.tf paths |
| docker-compose | Skill | ~/.claude/skills/ | Opus 4.6 | ~800 loaded | docker-compose* |
| powershell-style | Skill | ~/.claude/skills/ | Opus 4.6 | ~600 loaded | *.ps1 paths |
| security-reviewer | Agent | .claude/agents/ | Sonnet 4.6 | ~20K/run | T2+ escalation |
| quality-reviewer | Agent | .claude/agents/ | Sonnet 4.6 | ~15K/run | T2+ escalation |
| fixer | Agent | .claude/agents/ | Opus 4.6 | ~25K/run | After review |
| Bash blocker | Hook | ~/.claude/settings | N/A | ~0 | PreToolUse |
| Sensitive guard | Hook | ~/.claude/settings | N/A | ~0 | PreToolUse |
| Auto-format | Hook | ~/.claude/settings | N/A | ~0 | PostToolUse |
| Error recovery | Hook | ~/.claude/settings | N/A | ~0 | PostToolUseFail |
| Git attribution | Hook | ~/.claude/settings | N/A | ~0 | PostToolUse |
| Win notification | Hook | ~/.claude/settings | N/A | ~0 | Notification/ Stop |
| Turbo /finalize | Plugin | tobihagemann/ turbo | Opus 4.6 | ~30K/run | Manual |
| Turbo /review-code | Plugin | tobihagemann/ turbo | Opus 4.6 | ~40K/run | T2+ post-impl |
| OCR /ocr:review | Plugin | spencermarx/ocr | Host model | ~60K/run | T3 escalation |
|  | Plugin | openai/codex-cc | GPT-5.4 | ~15K/run | T3 escalation |

| Component | Type | Location | Model | Token Cost | Trigger |
|---|---|---|---|---|---|
| Codex / codex:review | | | | | |
| sequential-thinking | MCP | .mcp.json | N/A | ~0 idle | Complex tasks |
| github | MCP | .mcp.json | N/A | ~0 idle | PR/Issue ops |

# Part IV

Automation, Debugging & References

# Post-Implementation Automation Scripts

## new-project : Auto-Bootstrap Script

**What:** Creates .ocr/config.yaml, copies .mcp.json, symlinks global rules.
**Where:** `~/bin/new-project` (must be in PATH).
**When:** Run after cloning any new repository.
**Why:** Eliminates manual repetition : every new project starts fully configured.
**How:** Called manually or via the `clone()` shell function.

```
#!/bin/bash
# ~/bin/new-project : Run after cloning any new repo
PROJ_DIR=$(pwd)

# Create required directories
mkdir -p .ocr .claude/skills .claude/agents

# Symlink global rules into project
# (Rules at ~/.claude/rules/ are not detected in project dirs
#  without this symlink : discovered during implementation)
ln -sfn ~/.claude/rules .claude/rules 2>/dev/null

# Copy MCP config (sequential-thinking + github)
cp ~/.claude/.mcp.json .mcp.json 2>/dev/null

# Create OCR config with standard team composition
cat > .ocr/config.yaml << 'EOF'
team:
  principal: 2
  security: 2
  quality: 1
discourse:
  enabled: true
  rounds: 2
EOF

# Create .worktreeinclude for subagent worktree isolation
echo '.env' > .worktreeinclude
echo '.env.local' >> .worktreeinclude

echo "Project initialized at $PROJ_DIR"
```

```
chmod +x ~/bin/new-project
```

## clone() : Auto-Setup on Git Clone

**What:** Shell function wrapping git clone + auto-runs new-project.

**Where:** `~/.bashrc` (WSL) or `~/.zshrc` (Mac).

**Why:** Zero-friction project onboarding. Skips tool repos.

```
# Add to ~/.bashrc or ~/.zshrc
clone() {
  git clone "$@"
  local dir=$(basename "$1" .git)
  cd "$dir" || return
  case "$dir" in
    turbo|open-code-review|codex-plugin-cc) return ;;
  esac
  ~/bin/new-project
}
```

## check-project : Health Check Script

**What:** Displays checkmarks for all expected config files.

**Where:** `~/bin/check-project` .

**Why:** Quick diagnostic for debugging.

```
#!/bin/bash
# ~/bin/check-project : Health check for orchestration config
check() { [ -e "$1" ] && echo "  [OK] $1" || echo "  [!!] $1 MISSING"; }

echo "═══ Project: $(basename $(pwd)) ═══"
check .claude/settings.json
check .claude/rules
check .mcp.json
check .ocr/config.yaml
check .worktreeinclude
check CLAUDE.md
echo "═══ Global ═══"
check ~/.claude/CLAUDE.md
check ~/.claude/settings.json
echo "Skills: $(ls ~/.claude/skills/ 2>/dev/null | wc -l)"
echo "Agents: $(ls ~/.claude/agents/ 2>/dev/null | wc -l)"
```

## Complete Debugging Log : All 12 Errors Encountered

| # | Error | Phase | Root Cause | Fix |
|---|---|---|---|---|
| 1 | Model field "claude-opus-4-6[1m]" | P2 | ANSI bold escape code in config | Replace with exact "claude-opus-4-6" |
| 2 | autoMode.environment string | P9 | Must be array, not single string | Wrap in array: ["desc1", "desc2"] |
| 3 | "allowed-tools" in subagent | P7 | Wrong field name (skills use allowed-tools) | Use "tools" for subagents |
| 4 | "memory-scope" in subagent | P7 | Wrong field name | Use "memory" for subagents |
| 5 | "codex auth login" | P6 | No "auth" subcommand exists | Use "codex login" |
| 6 | Threshold not firing T1/T2 | P3 | Skill not auto-loading every turn | Add mandatory section to CLAUDE.md |
| 7 | Rules path not detected | Post | ~/.claude/rules/ not seen in projects | Symlink into each project .claude/ |
| 8 | Git attribution quoting | Post | Bash inline script escaping issues | Use Python script instead |
| 9 | GitHub push protection | Post | Hardcoded OAuth token in .mcp.json | Use ${GH_TOKEN} env var |
| 10 | Portfolio settings.json | Post | // comments + terminal artifacts | Delete corrupted file |
| 11 | Turbo wrote to global CLAUDE.md | P4 | Appended to ~/.claude/CLAUDE.md | Restored, then re-added (useful) |
| 12 | /plugin add opened browser | P4 | Turbo is skills, not traditional plugin | Use /plugin marketplace add |

# References and Citations


1. Kim, Y. et al. (2025). Multi-Agent Orchestration: Performance Boundaries and Architecture Selection. Google Research/DeepMind/MIT. arXiv:2512.08296.
2. MDAgents (2024). Adaptive Complexity-Based Multi-Agent Collaboration. NeurIPS 2024 Oral, MIT Media Lab.
3. Costa, A. (2026). AgentSpawn: Selective Activation and Adaptive Spawning. arXiv:2602.07072.
4. Kaesberg, J. et al. (2025). Voting Improves Reasoning in Multi-Agent LLM Systems. ACL 2025, University of Göttingen.
5. Cemri, M. et al. (2025). Multi-Agent Framework Failure Analysis. UC Berkeley. NeurIPS 2025 Spotlight.
6. Zhu, J. et al. (2025). Topology Effects on Multi-Agent Coordination. ACL 2025.
7. JetBrains AI Pulse Survey (2026). 10,000+ developers. January 2026.
8. Stack Overflow Developer Survey (2025). 49,000+ respondents.
9. DX Q4 2025 Impact Report. 135,000+ developers.
10. Faros AI Productivity Paradox Report (2025). 10,000+ developers, 1,255 teams.
11. METR Randomized Controlled Trial (2025). 16 experienced developers, 246 tasks.
12. GitHub/Accenture Copilot Enterprise Study (2024). Enterprise acceptance rates.
13. Cursor Bugbot (2025). 40% review time reduction across 1M+ PRs.
14. Greptile Bug Detection Benchmark (2025). 50 real production bugs.
15. Ghost Security SAST Study (2025). ~3,000 repositories analyzed.
16. GitClear Code Churn Analysis (2024-2025). Churn rising from 3.3% to 5.7-7.1%.
17. McConnell, S. Code Complete (2nd ed.). Human code inspection catch rates: 55-60%.
18. DORA 2025 Report. 15% improvement in bug detection for high-performing teams.
19. CodeRabbit Analysis (2025). AI-generated code produces 1.7x more issues.
20. Anthropic Claude API Pricing (2026). Opus 4.6: $5/$25, Sonnet 4.6: $3/$15, Haiku 4.5: $1/$5.
21. Anthropic MCP Donation to Linux Foundation (2025). Agentic AI Foundation.
22. Claude Code Best Practices (2026). code.claude.com/docs/en/best-practices.
23. HumanLayer CLAUDE.md Analysis (2025). ~150-200 instruction budget.
24. Addy Osmani (2026). The Code Agent Orchestra. addyosmani.com.
25. oh-my-claudecode (2026). Multi-agent orchestration. 3-5x speedup, 30-50% token savings.
26. CB Insights (2025). AI Coding Tools Market share analysis.
27. McKinsey AI Productivity Survey (2025). ~300 publicly traded companies.
28. Gartner AI Code Assistant Forecast (2025). 90% of enterprise developers by 2028.
29. Forrester (2025). 80% enterprise teams using generative AI by 2026.

# Appendix D: Config File Precedence

| Scope | Location | Precedence | Git? |
|---|---|---|---|
| Managed (highest) | managed-settings.json | Cannot be overridden | N/A |
| CLI arguments | --flag at launch | Session-only override | No |
| Local | .claude/settings.local.json | Overrides Project+User | No |
| Project | .claude/settings.json | Overrides User | Yes |
| User (lowest) | ~/.claude/settings.json | Base defaults | No |

Hooks MERGE across all levels (all fire). CLAUDE.md is ADDITIVE.
MCP servers: local overrides project overrides user.

# Appendix E: Subagent Frontmatter Reference

| Field | Required | Notes |
|---|---|---|
| name | Yes | Unique identifier |
| description | Yes | Drives auto-selection |
| tools | No | Allowed tools (NOT allowed-tools) |
| disallowedTools | No | Denied tools |
| model | No | sonnet, opus, haiku, full ID, or inherit |
| memory | No | user, project, local (NOT memory-scope) |
| isolation | No | worktree for git worktree isolation |
| effort | No | low, medium, high, max (Opus only) |
| maxTurns | No | Max agentic turns |
| skills | No | Skills to preload |
| mcpServers | No | MCP server access |
| hooks | No | Scoped lifecycle hooks |
| background | No | Run as background task |
| initialPrompt | No | Auto-submitted first turn |

Model resolution: CLAUDE_CODE_SUBAGENT_MODEL env → per-invocation → frontmatter → main model

# Appendix F: Token Economics

Only officially documented or verified claims included. Community observations labeled as such.

| Metric | Value | Source |
|---|---|---|
| Cache TTL | 5 min default, 1hr optional (all tiers) | Official API docs |
| Max subscription | 1hr TTL absorbed into pricing | Official pricing |
| Cache read cost | 0.1x base input price | Official API docs |
| Cache write cost (5min) | 1.25x base input | Official API docs |
| Cache write cost (1hr) | 2x base input | Official API docs |
| System prompt | ~20K tokens | Community measurement |
| CLAUDE.md target | Under 200 lines per file | Official best practices |
| Skill description | ~250 chars truncation | Official docs |
| Cache hit rate | ~90%+ typical (varies) | Community observation |
| 5-hour rolling window | Confirmed rate limit | Official docs |
| 7-day ceiling | NOT confirmed | Community inference only |
| Opus vs Sonnet cost | Opus ~1.7x more against limits | Community measurement |

30. Microsoft WSL 2 Performance (2024). 87-95% native. Cross-filesystem at 6%.
31. shanraisshan/claude-code-best-practice (2026). Settings reference. GitHub.