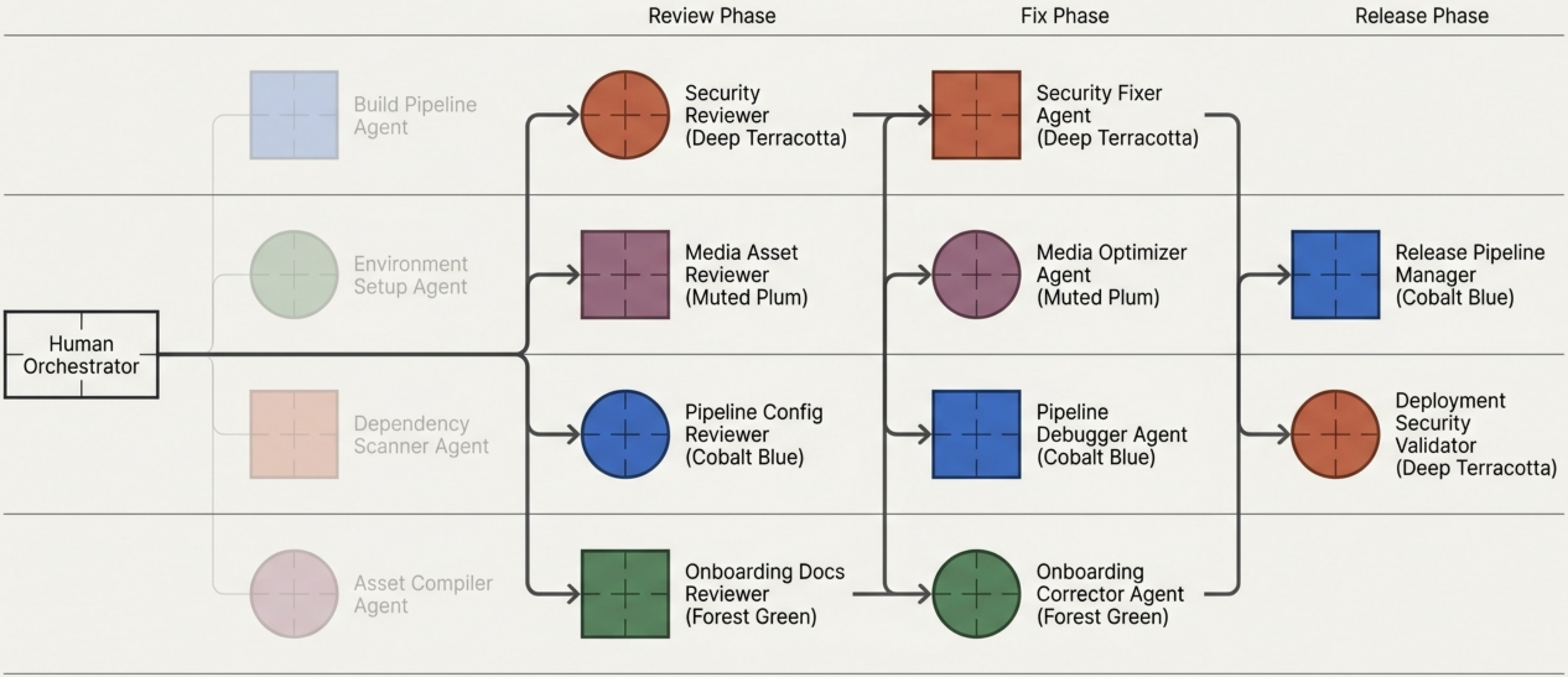
Shipping with Claude Code

Four Reviewers, Three Fixers, One Release Pipeline



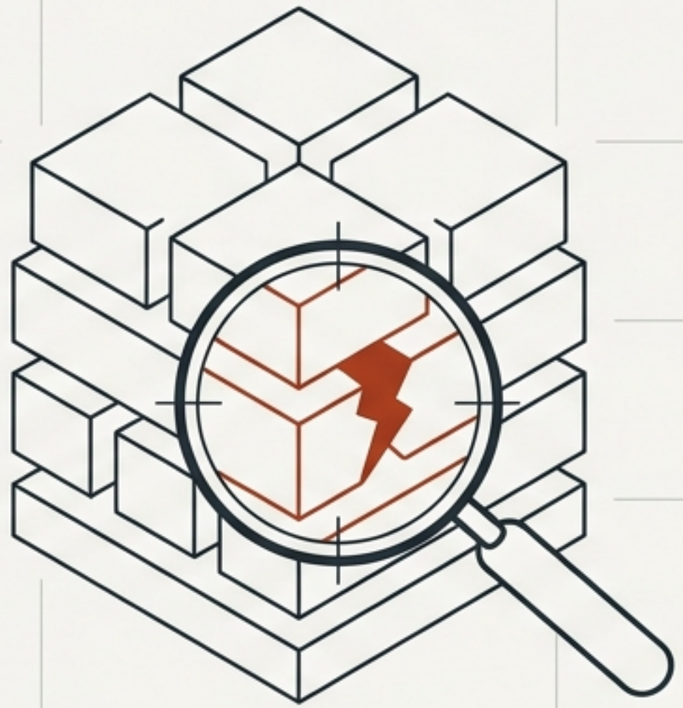
The Baseline: “It Works” vs. “It Holds Up”

Building is only half the job. Code that functions perfectly can still harbor deep architectural or security flaws.

The Build

81	Commits
5	Parallel Agents
3	Architectural Layers
✓	All Typechecks Passed.

The Reality Check



The Silent Failure: Prompt Leaks

Dead template placeholders bypassed the onboarding replacement logic. The model ignored the gibberish and generated fine responses, but wasted tokens on every single message.

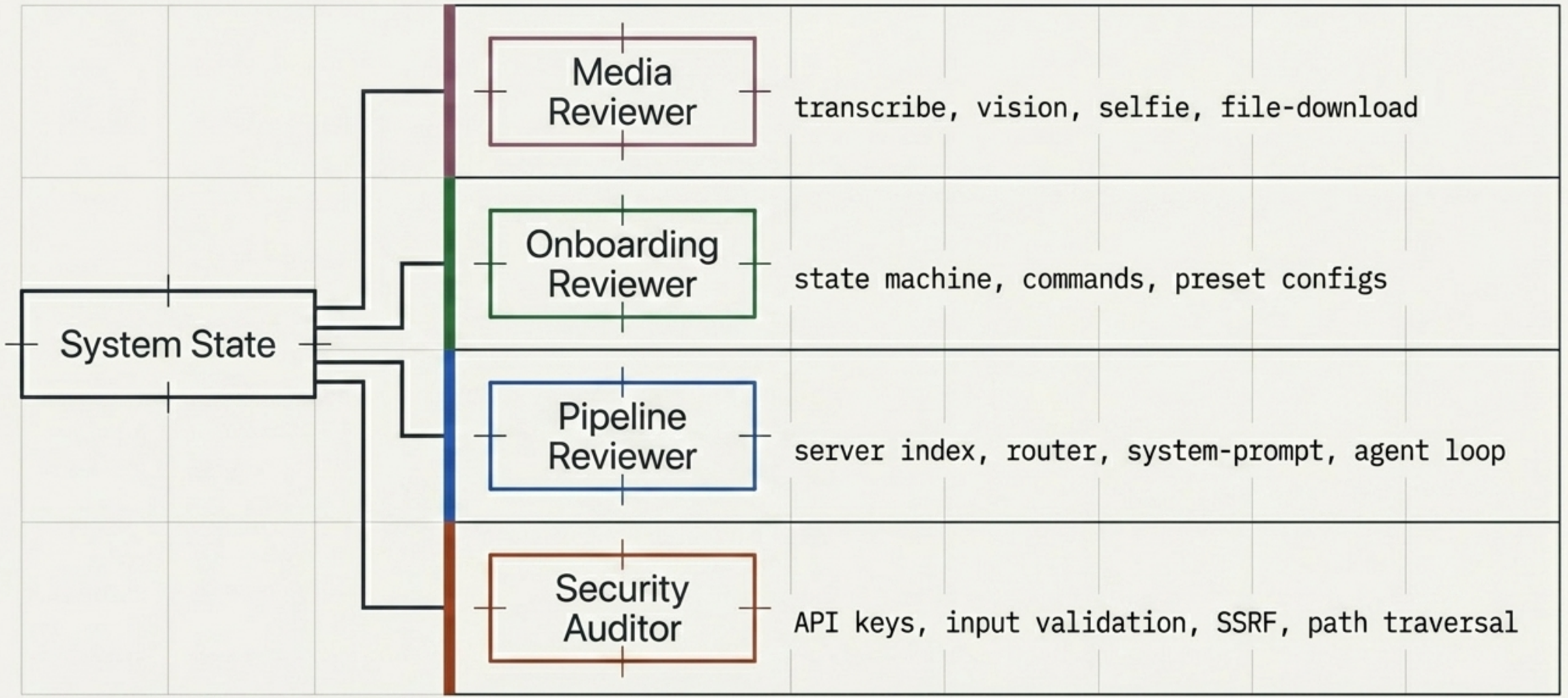
```
SYSTEM_PROMPT_CONFIG = {
  "role": "system",
  "content": "You are a helpful assistant. {custom_story}"
  User instructions..."
  ...
}

"role": "user",
  "messages": [
  "content": "system ",
  "custom_story":"Summarize the context ..."
}
```

Working code can still waste tokens.

LLM Payload

```
{
  "messages": [
    {
      "role": "system",
      "content": "You are a helpful assistant.
  Context: {custom_story} User instructions..."
    },
    {
      "role": "user",
      "content": "Summarize the context."
    }
  ]
}
```

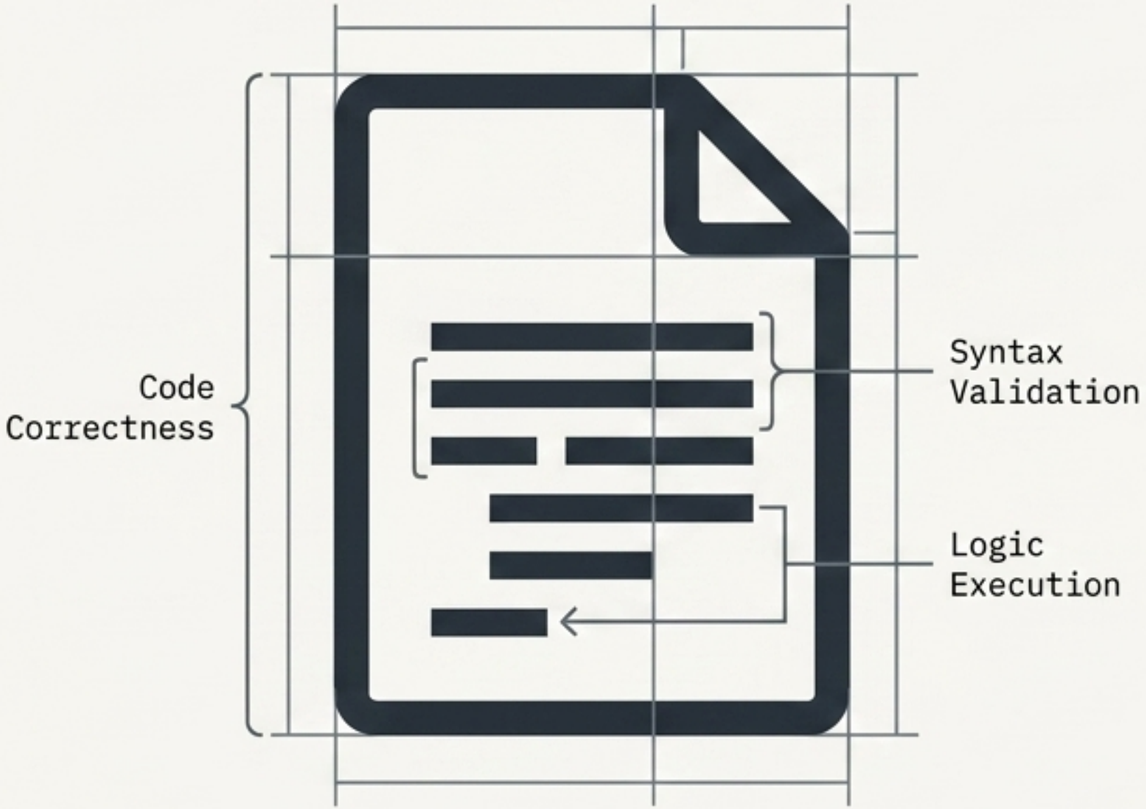
Orchestrating Adversarial Review



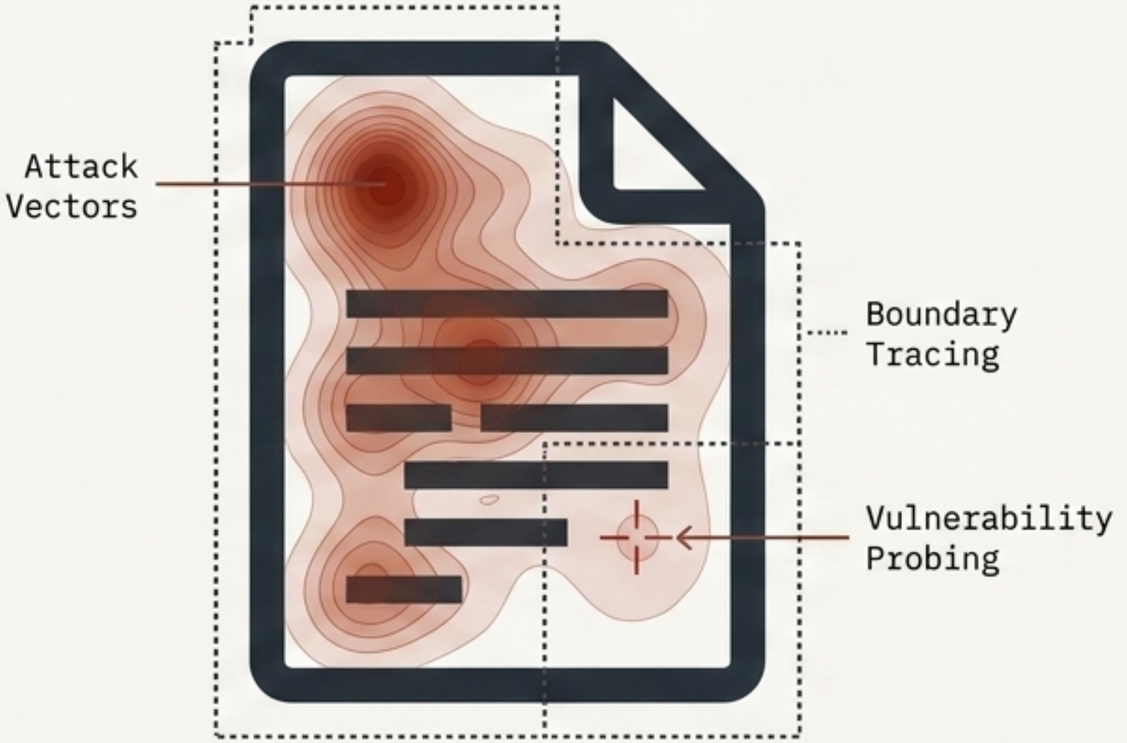
The Value of Specialization

They look at the exact same code and see entirely different things.

The Generalist



The Security Specialist



The Five-Minute Review Audit

REVIEWER	VERDICT	CRITICAL	HIGH	MEDIUM
Media	WARNING	0	4	10
Pipeline	BLOCK	2	4	6
Onboarding	WARNING	0	4	7+
Security	BLOCK	2	4	5

> Two BLOCKs. Two WARNINGs. Not shippable.

Tracing the Critical Vulnerabilities

Missing Cost Attribution

processMedia() called before userId availability, resulting in untrackable costs.

Typing Indicator Leak

Missing clearInterval on empty message batch causes indefinite bot typing state.

Path Traversal in HTTP API

/api/agents passed unvalidated presetId directly to file path.

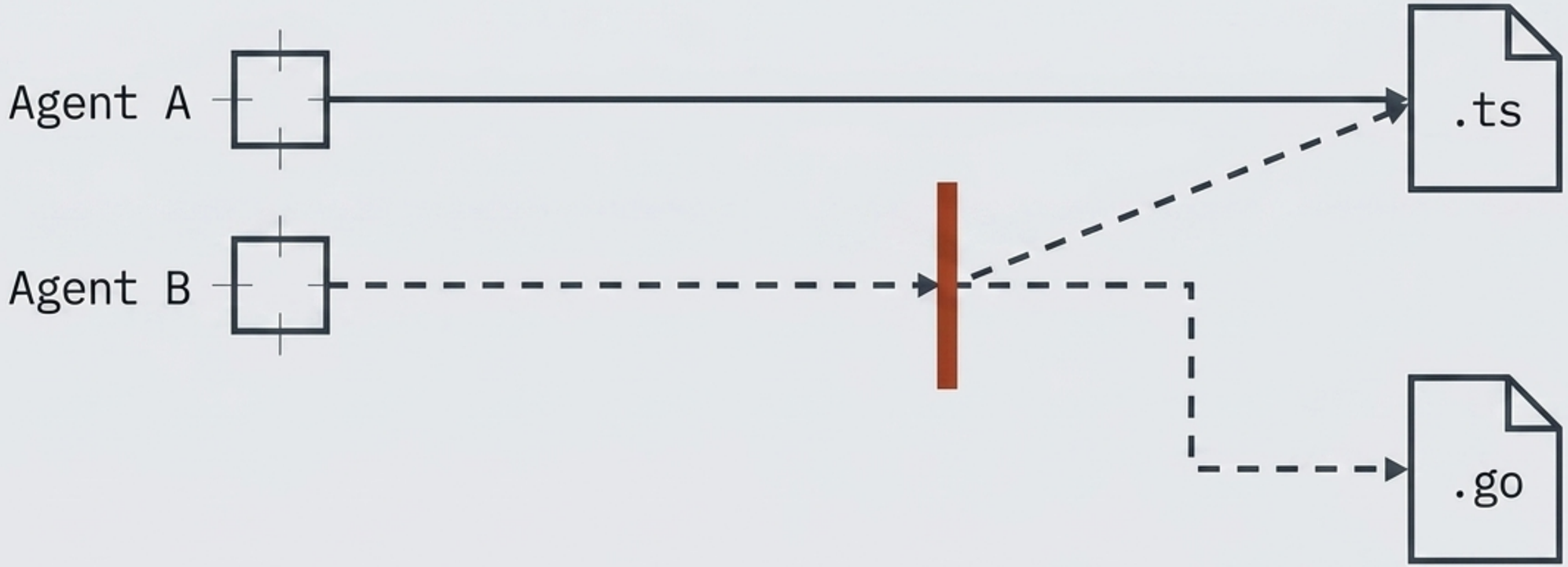
Bot Token Leak

Telegram download URLs leak raw tokens into error logs upon failure.

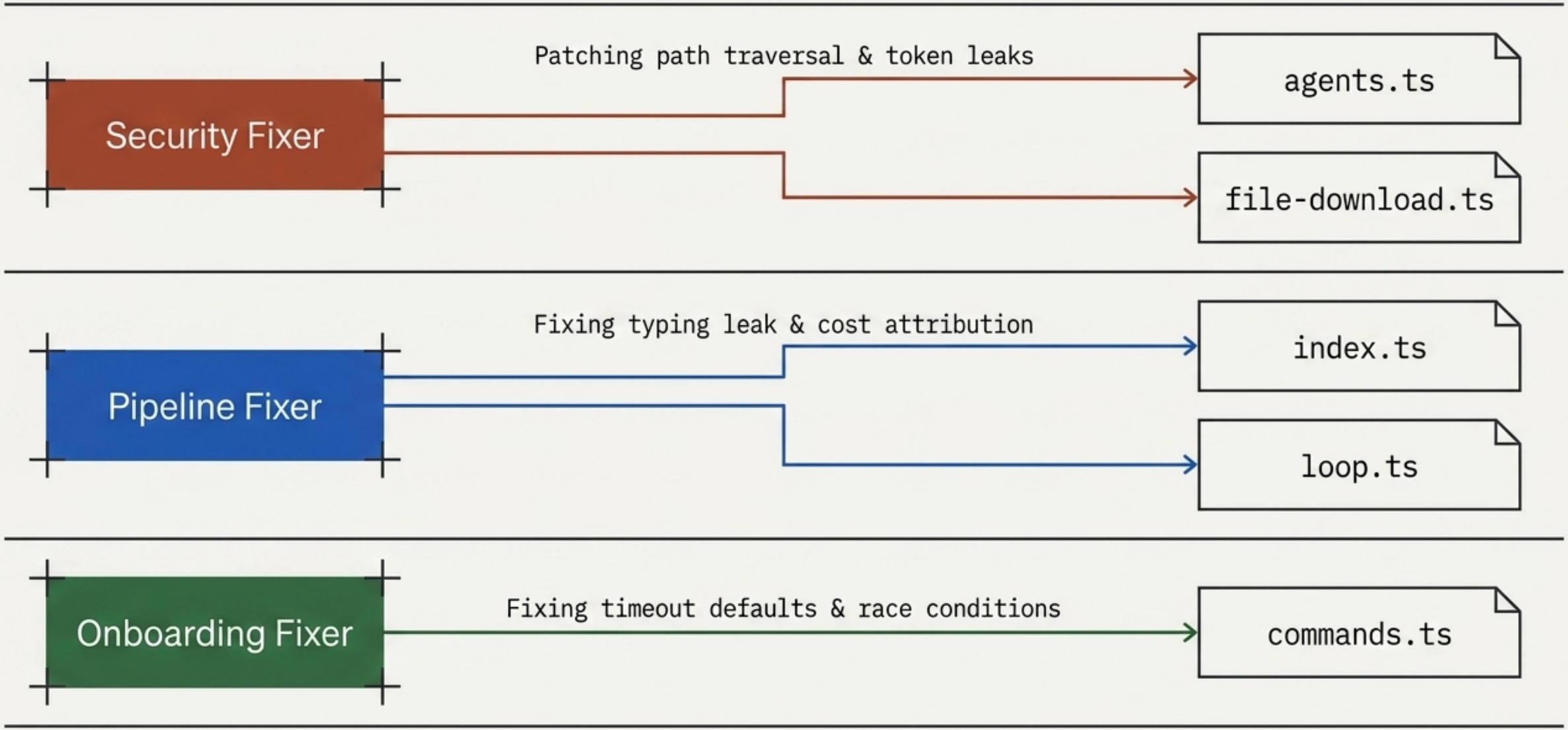
The Golden Rule of Parallel Fixing

No two agents edit the same file.

Split responsibilities by file, not by issue. If two issues touch the same file, they belong to the same fixer. This eliminates merge conflicts entirely.

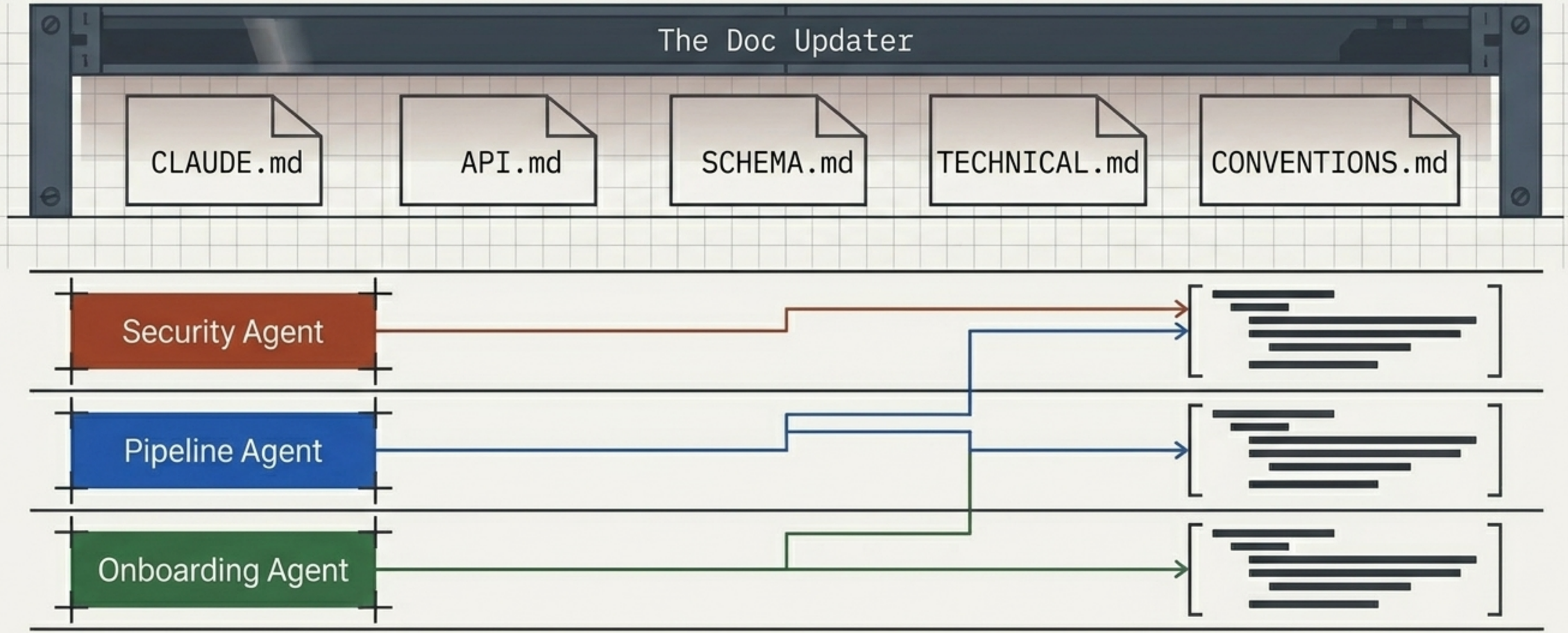


Routing the Fix Agents



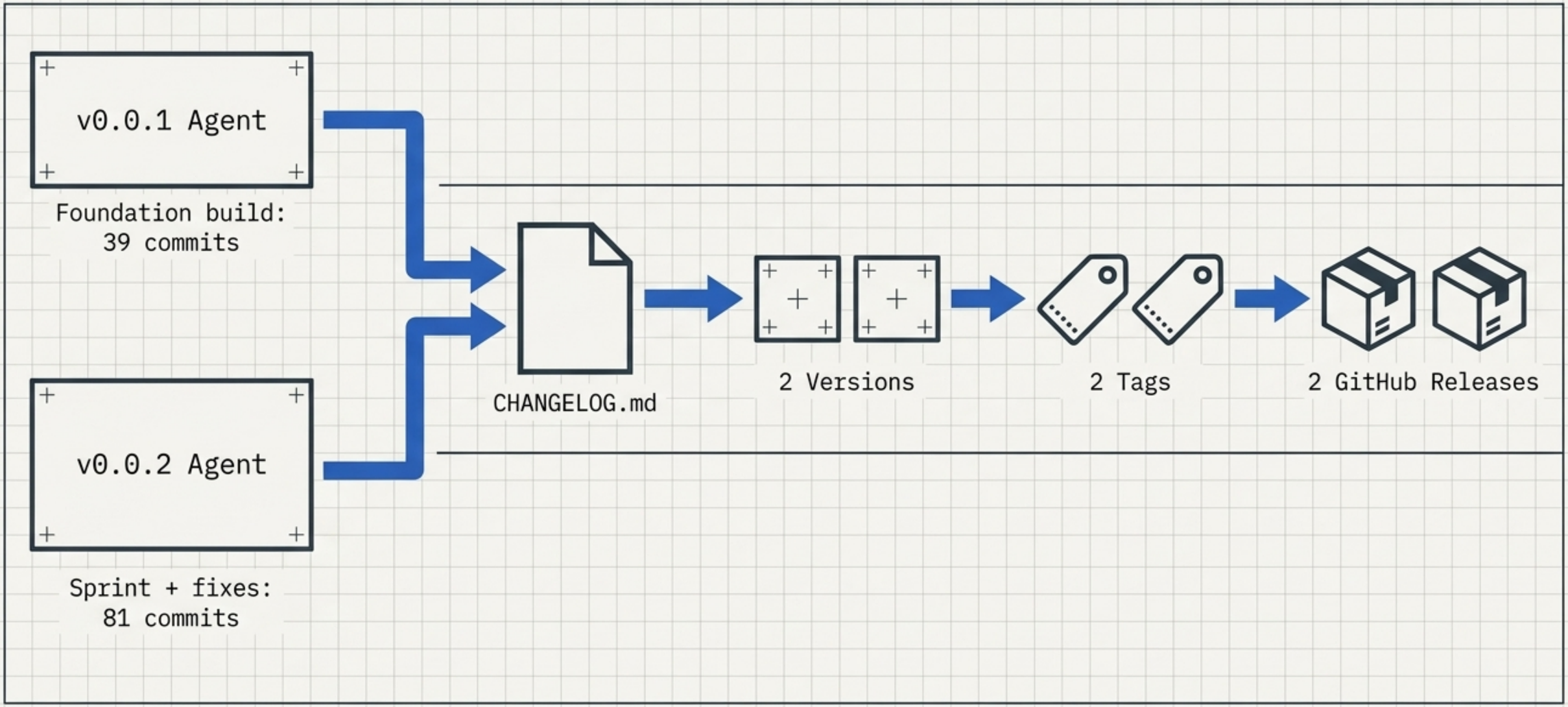
Combating Documentation Drift

Documentation drift is real. A doc-updater agent running in parallel costs nothing and prevents stale docs later.



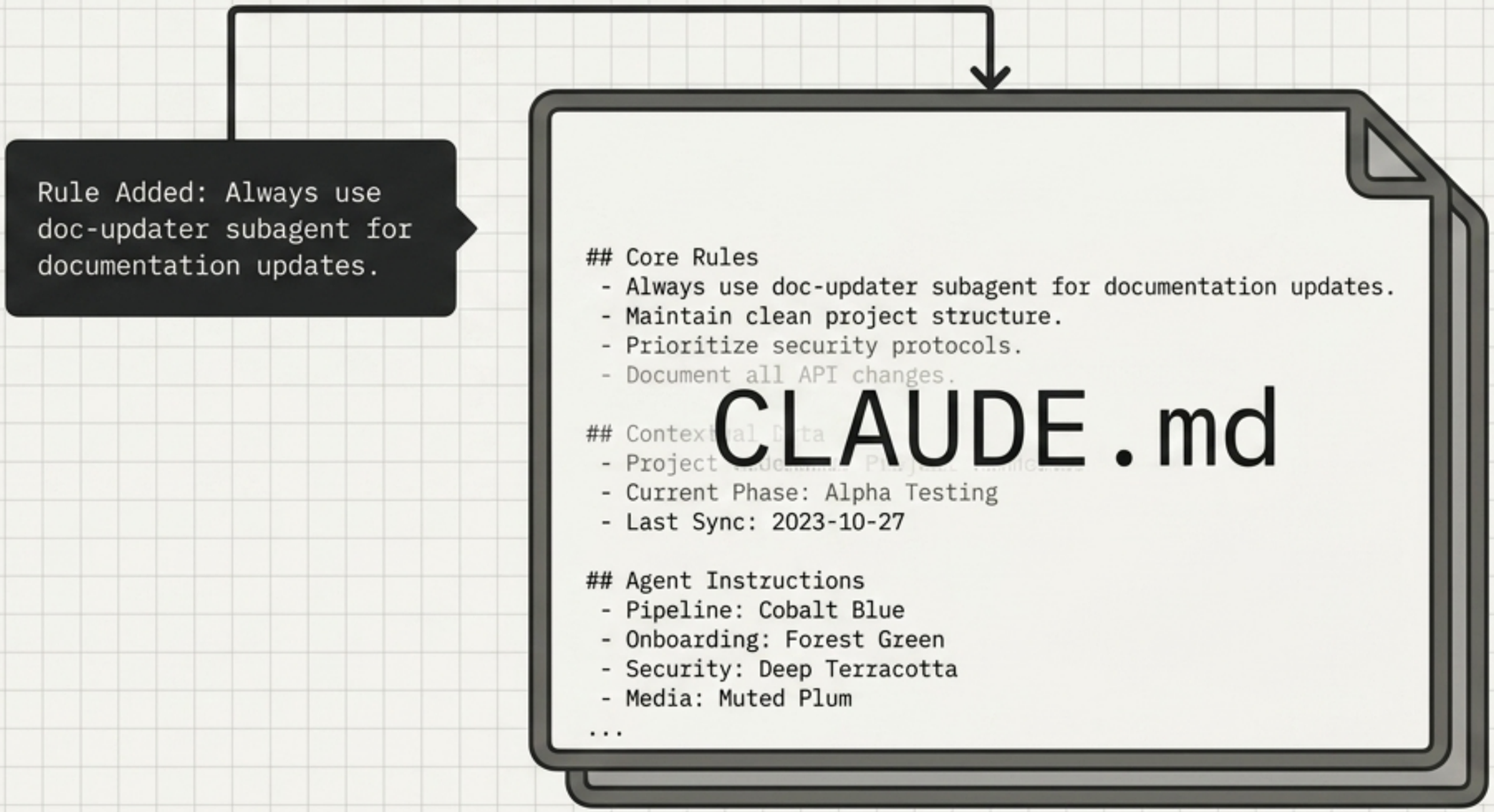
Automating the Release Ceremony

Delegating mechanical tasks that don’t benefit from human judgment.



Encoding Institutional Memory

What goes into CLAUDE.md persists across sessions, context compactions, and crash recoveries. Institutional memory compounds.

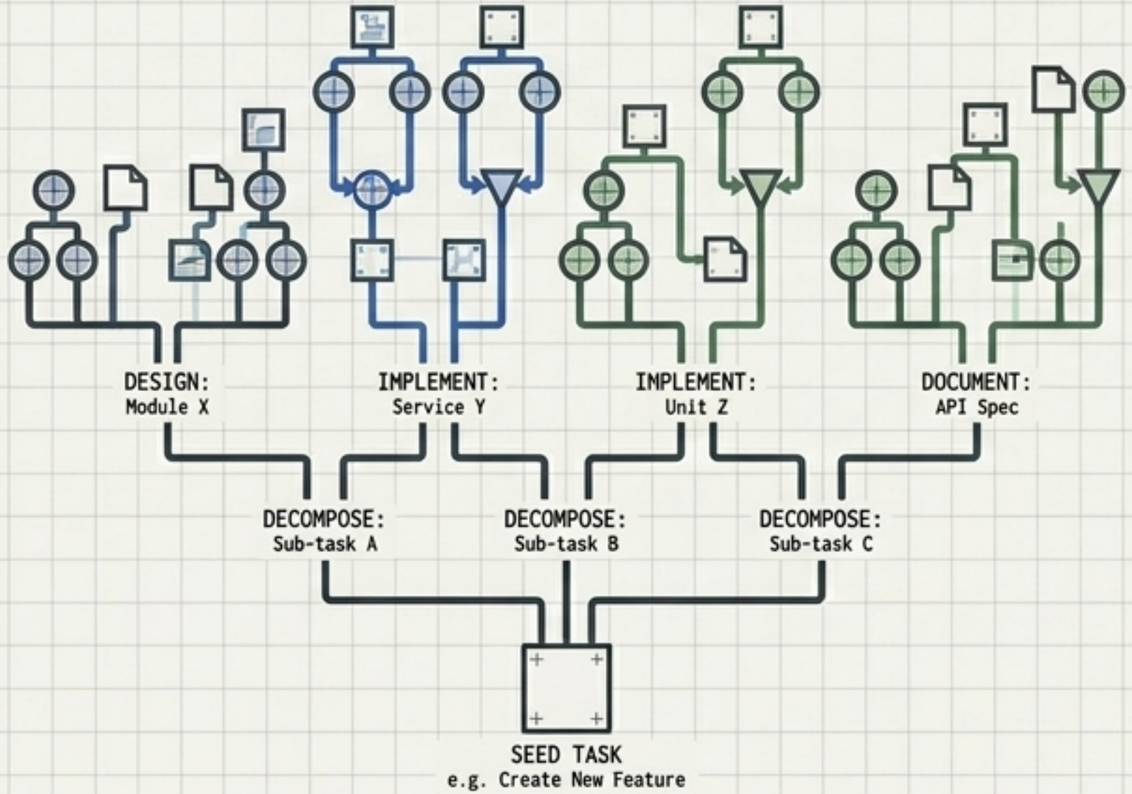


Two Halves of the Same Workflow

Opposite shapes of work powered by the exact same mechanics: task decomposition, file ownership, and parallel execution.

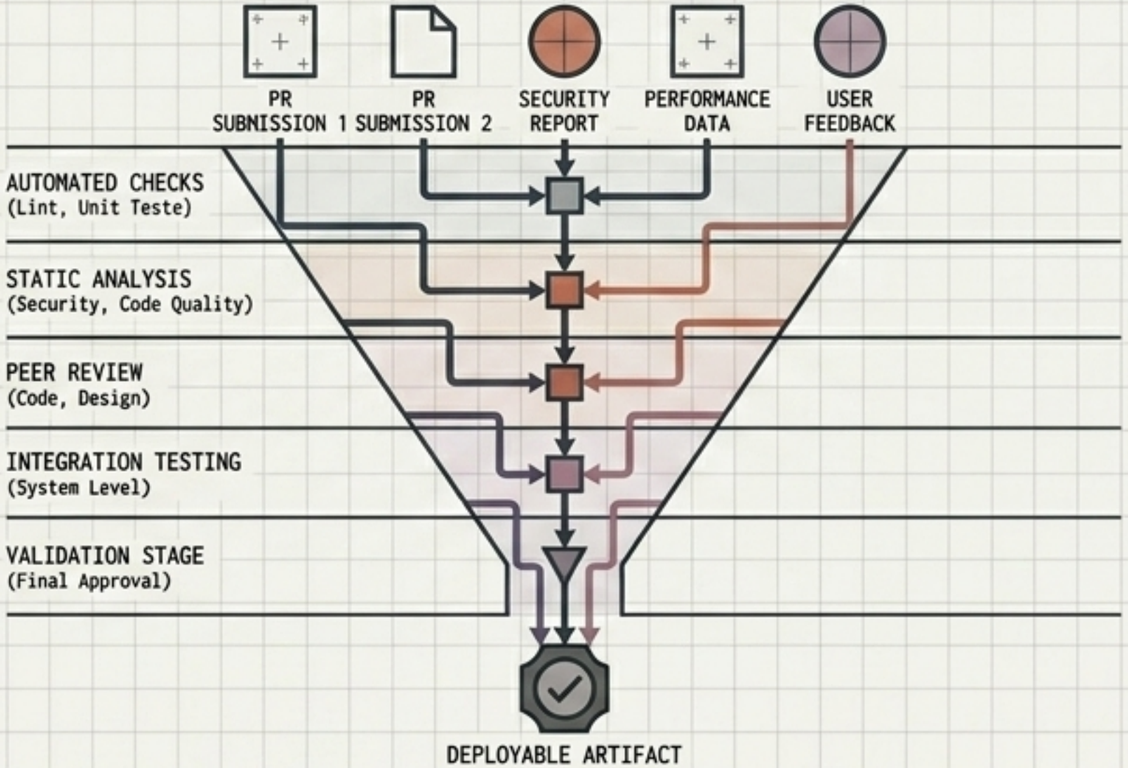
Building

Shape of work: Creative, Forward-looking, Generative.



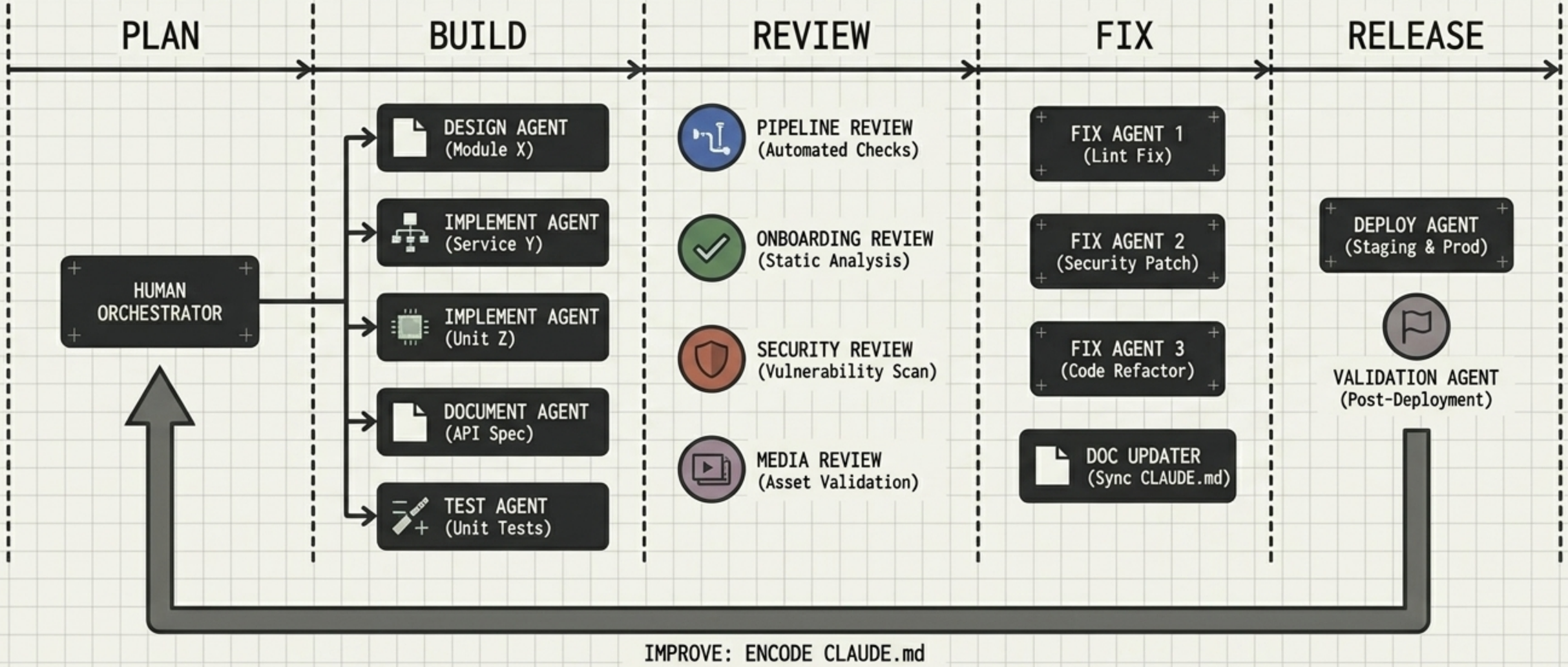
Reviewing

Shape of work: Adversarial, Retrospective, Reductive.



The Complete Cycle: 1 Session, 14 Agents

Not just building with AI. Shipping with AI.



The 6 Rules for Multi-Agent Workflows

1. Specialize Reviewers

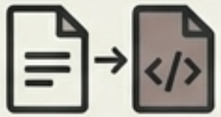
Domain lenses catch what generalists miss.

2. Strict File Ownership

Split responsibility by file to **eliminate** merge conflicts.

3. Fix in Parallel

Coordination overhead **is trivial** compared to time saved.

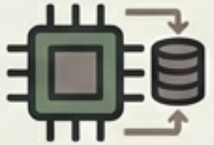


4. Run Parallel Doc Updates

Update docs simultaneously with code fixes.

5. Automate Ceremony

Delegate changelogs, tags, and releases.



6. Encode Memory

Write process improvements into `CLAUDE.md` immediately.