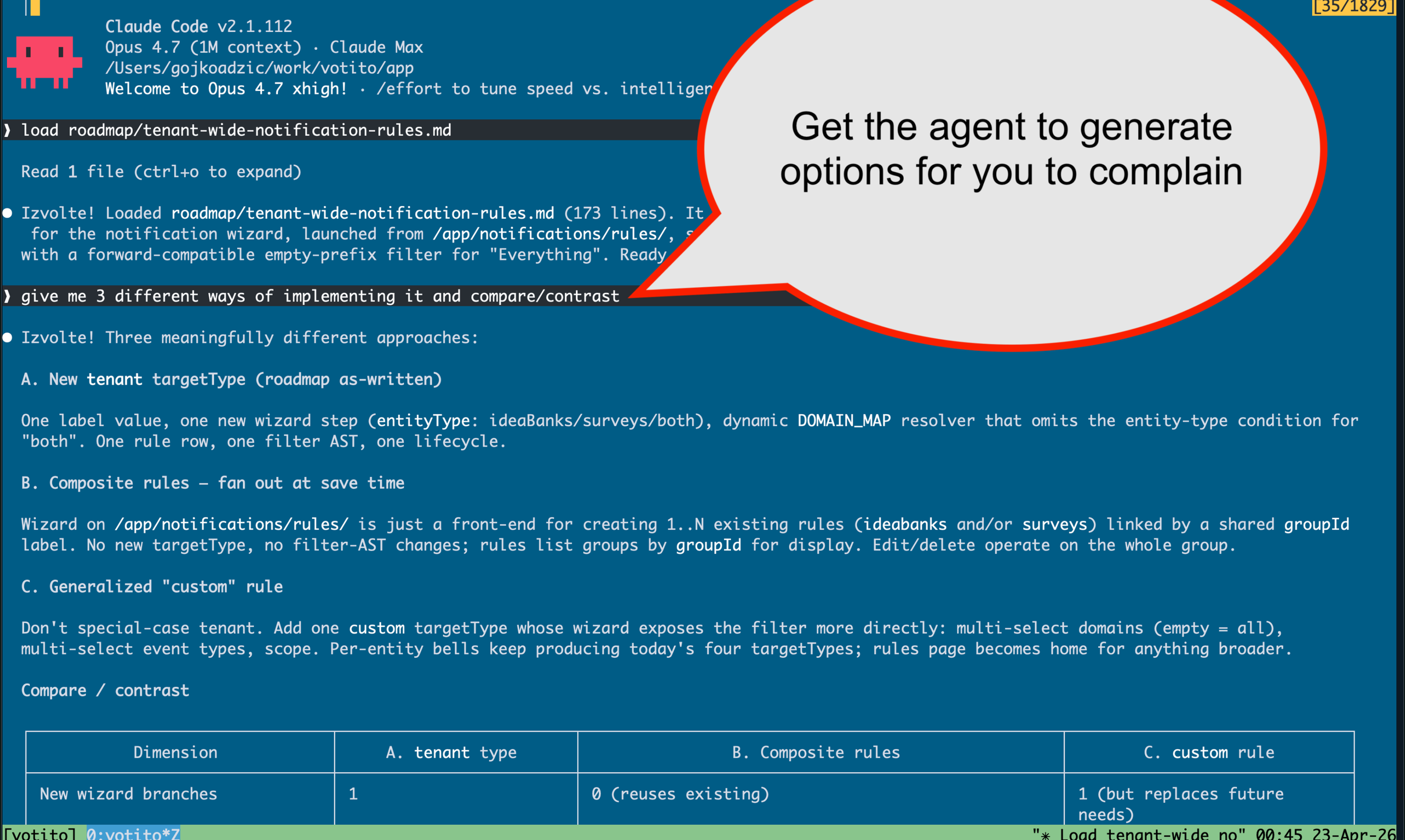
Votito

HOW TO GROW YOUR OWN SOFTWARE FACTORY

Gojko Adzic, gojko@votito.com

PEOPLE CAN
COMPLAIN MUCH
EASIER THAN THEY
CAN EXPLAIN.



Get the agent to generate options for you to complain

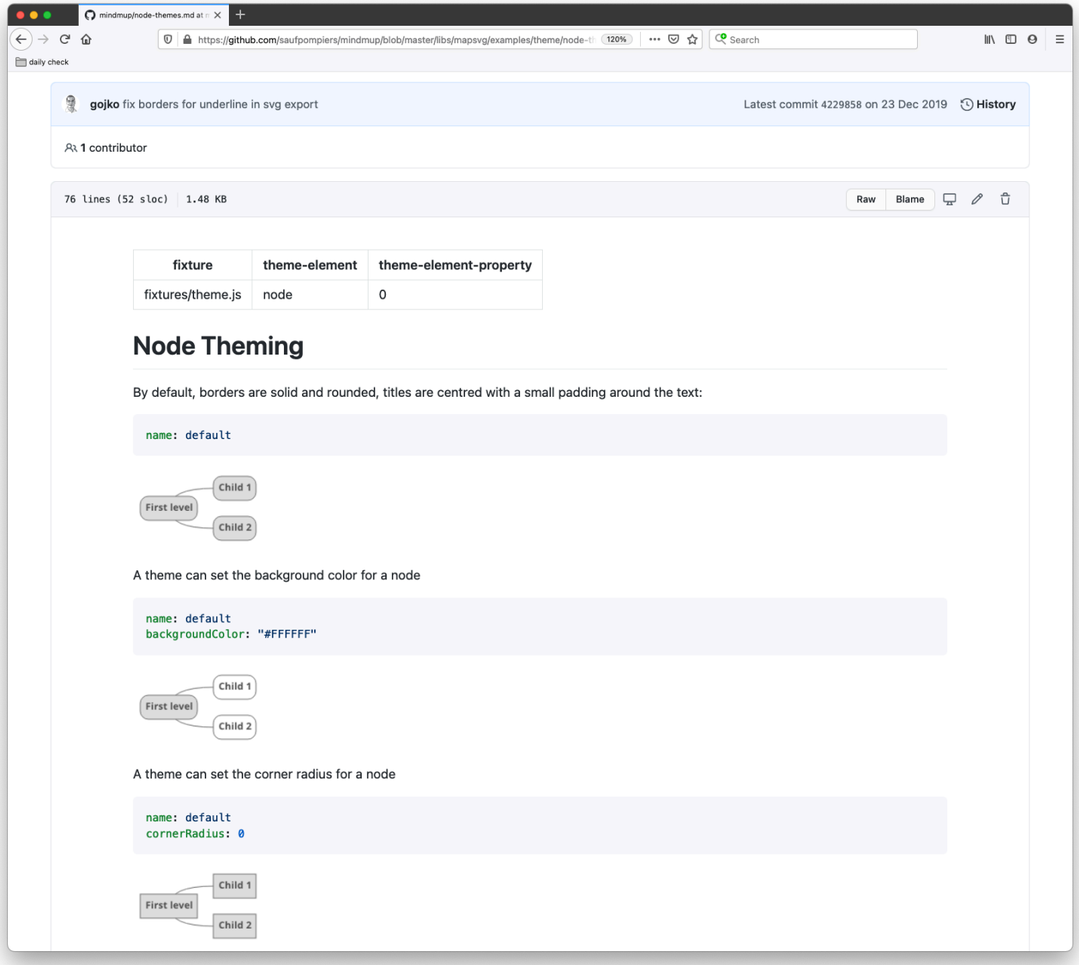
NOW WE HAVE INFINITE INTERNS THAT CAN WRITE ANYTHING FOR YOU, BUT YOU HAVE TO CHECK.

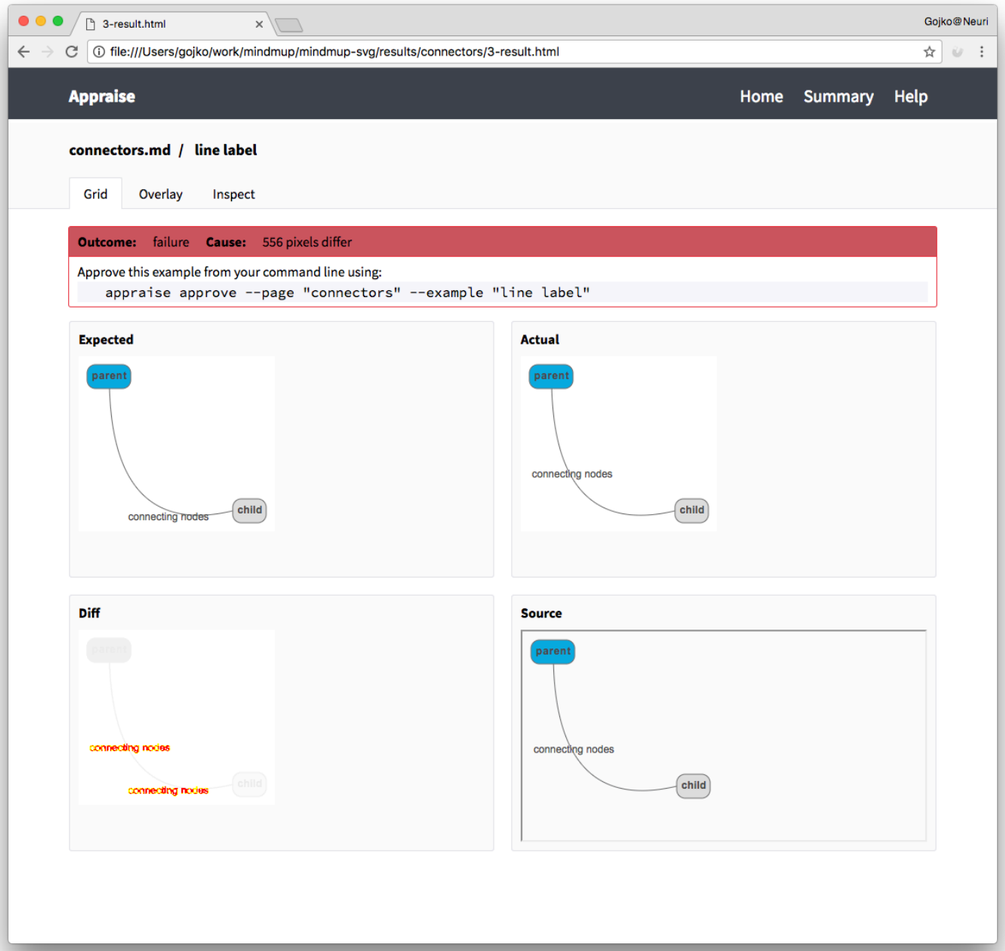
– BEN EVANS

https://www.ben-evans.com/benedictevans/2023/7/2/working-with-ai

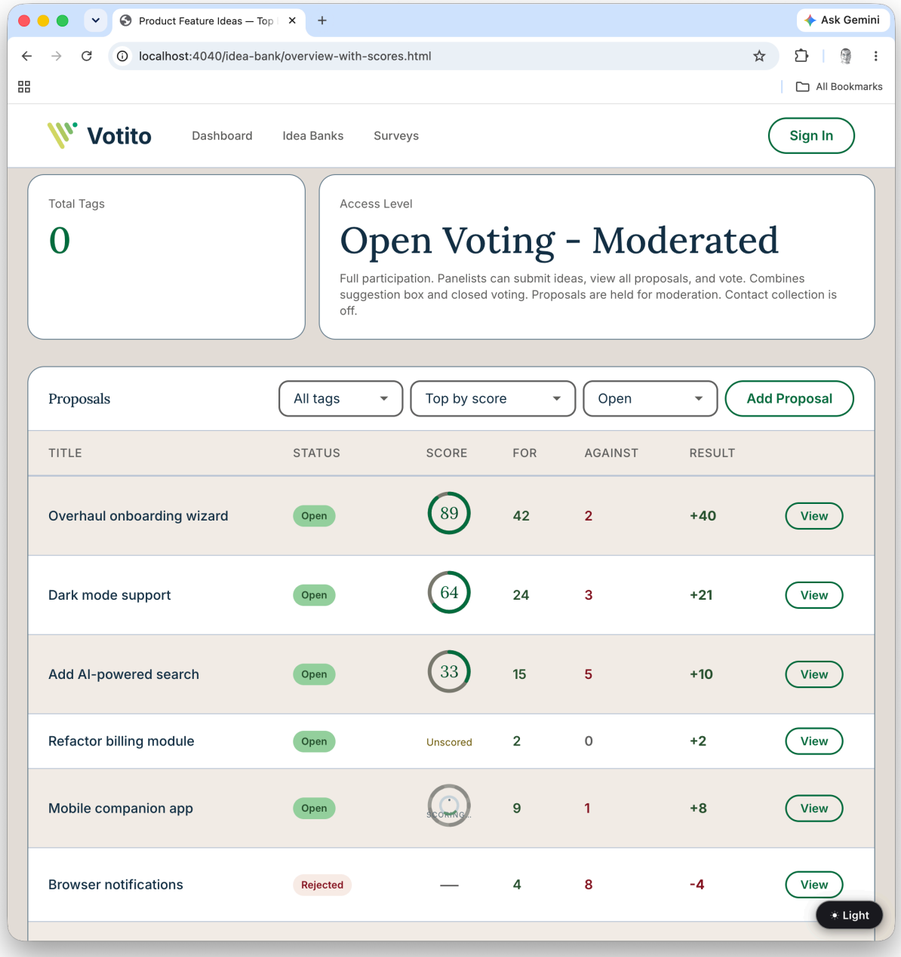
HUMAN REVIEW IS THE NEW BOTTLENECK

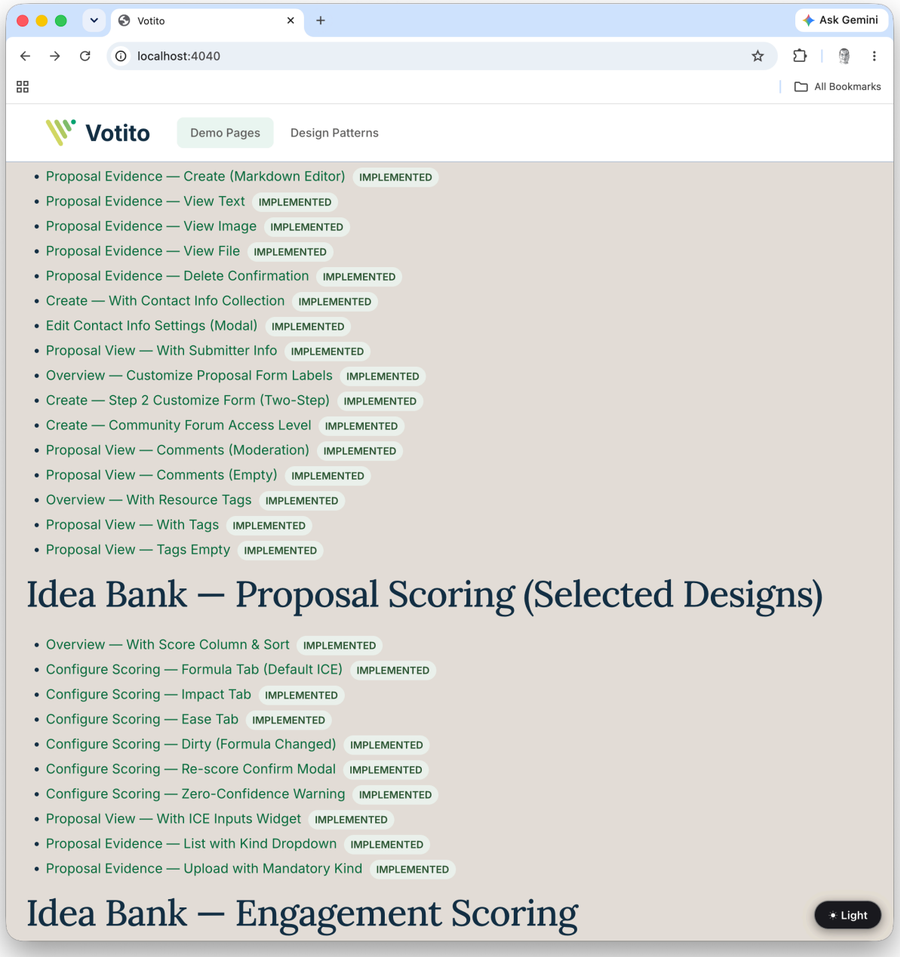
BIG IDEA IN 2018





TINY IDEA IN 2026

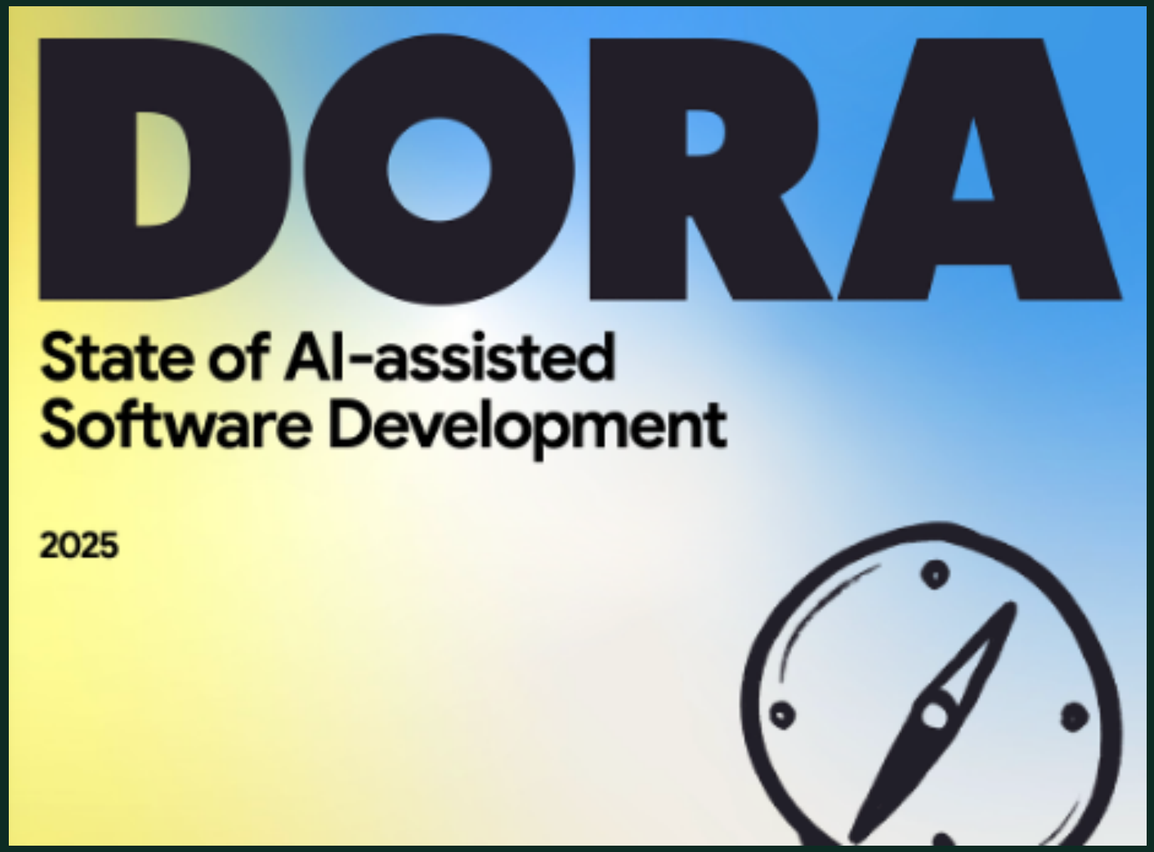




Nearly all *developers* (99%) *save time* using AI tools.



AI adoption has a *negligible effect on organizational performance* when *platform* quality is *low*, but when platform quality is *high*, the effect is *strong and positive*

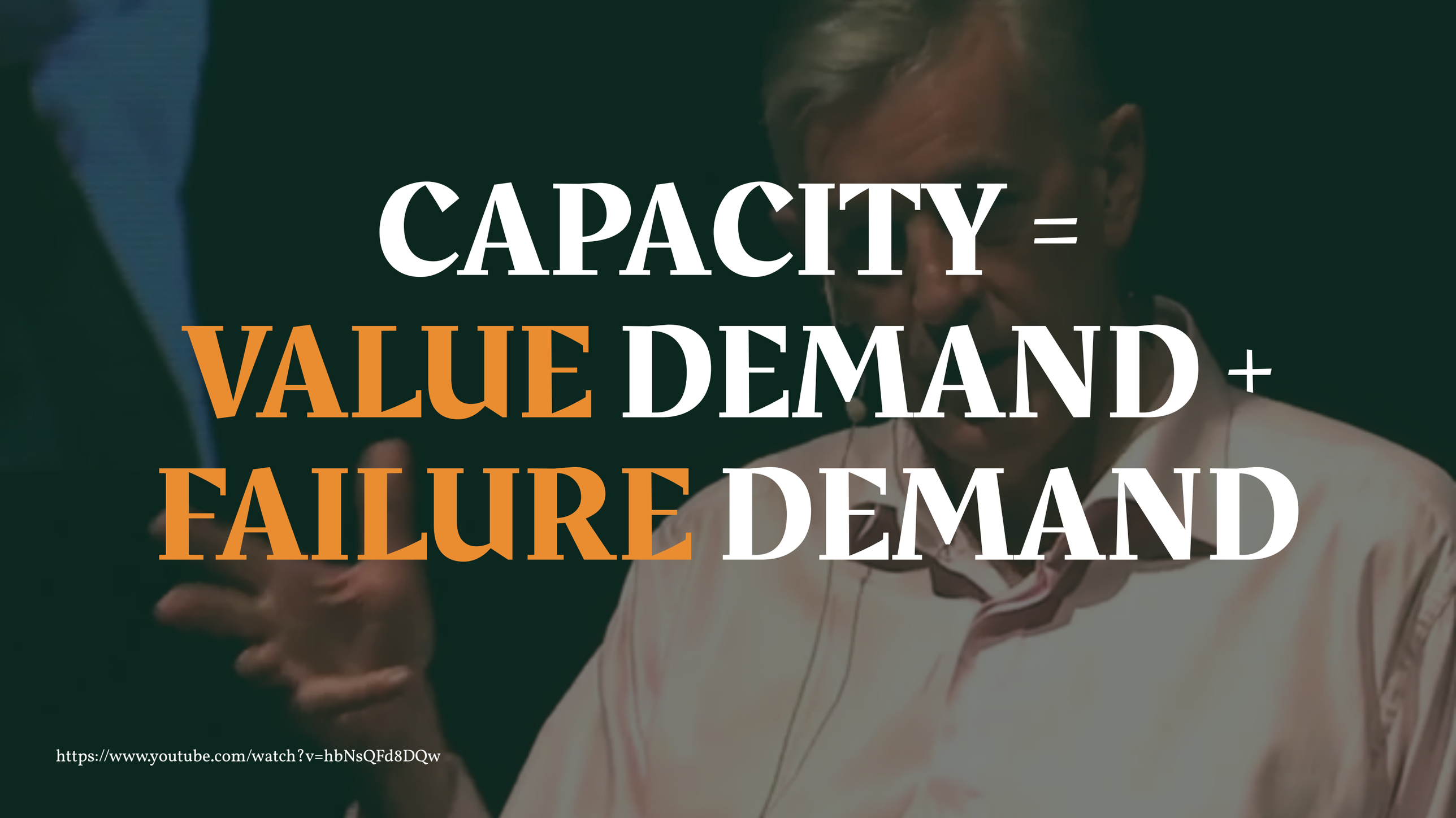


DARK FACTORY

- fully automated
- vibe coded
- **agents orchestrate** agents
- **markdown prayers**
- humans **evaluate at the end**
- magic

LIGHTS-ON FACTORY

- high degree of automation
- spec/plan driven
- **platform** orchestrates
- **deterministic guardrails**
- humans **supervise**
- engineering



https://www.youtube.com/watch?v=hbNsQFd8DQw


1. UNDERSTAND
2. IMPROVE
3. PULL IT
https://www.youtube.com/watch?v=hbNsQFd8DQw

IMPROVE HOW AGENTS MAKE DECISIONS

- boundaries (**reduce options**)
- self feed-back (**better conclusions**)
- cyclic feed-forward (**avoid re-learning**, faster decisions)
- decision visibility (**humans verify/supervise**)

OUR LIGHTS ON DECISION REVIEW

INTENT ➤ IMPACT ➤ GALLERY ➤ SPEC/PLAN ➤ SOFTWARE ➤ DEPLOY

(example, your process may differ)

Post



Slobodan Stojanović
@slobodan_

My new personal Claude Code record: 207 agents in 30 waves (+ gate, acceptance, fixes, and write-up). And it worked!

~3 hours of planning, 2 interventions, ~100 agents ran while I slept, and then it needed my input.

7:31 PM · Sep 24, 2026 · **803** Views

LIGHTS-ON FACTORY

1. container
2. workflow
3. regulator
4. orchestrator
5. runtime (optional)



version v6.2.1 License MIT node >=20.0.0 Discord Join Community

Build More Architect Dreams — An AI-driven agile development module for the BMad Method Module Ecosystem, the best and most comprehensive Agile AI Driven Development framework that has true scale-adaptive intelligence that adjusts from bug fixes to enterprise systems.

100% free and open source. No paywalls. No gated content. No gated Discord. We believe in empowering everyone, not just those who can pay for a gated community or courses.

Why the BMad Method?

Traditional AI tools do the thinking for you, producing average results. BMad agents and facilitated workflows act as expert collaborators who guide you through a structured process to bring out your best thinking in partnership with the AI.

- **AI Intelligent Help** — Invoke the `bmad-help` skill anytime for guidance on what's next
- **Scale-Domain-Adaptive** — Automatically adjusts planning depth based on project complexity
- **Structured Workflows** — Grounded in agile best practices across analysis, planning,



🌱 Spec Kit

Build high-quality software faster.

An open source toolkit that allows you to focus on product scenarios and predictable outcomes instead of vibe coding every piece from scratch.



Table of Contents

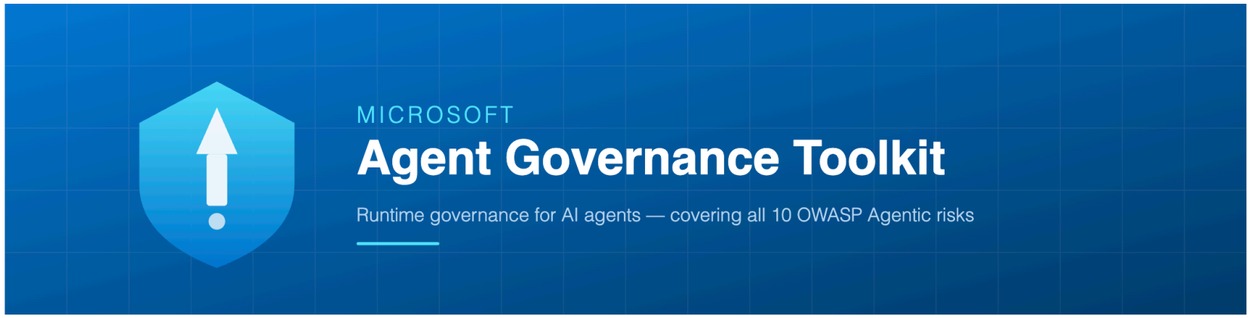
- 🤔 What is Spec-Driven Development?
- ⚡ Get Started
- 📽️ Video Overview
- 🚶 Community Walkthroughs
- 🤖 Supported AI Agents
- 🔧 Specify CLI Reference
- 🧩 Making Spec Kit Your Own: Extensions & Presets
- 📚 Core Philosophy
- 🌟 Development Phases
- 🎯 Experimental Goals
- 🔧 Prerequisites
- 📖 Learn More
- 📋 Detailed Process

GitHub - microsoft/agent-gov

github.com/microsoft/agent-governance-toolkit

README | Code of conduct | Contributing | MIT license | Security

English | 日本語 | 简体中文 | 한국어



Agent Governance Toolkit

Ship agents to production without losing sleep

FULL DOCUMENTATION | MICROSOFT.GITHUB.IO/AGENT-GOVERNANCE-TOOLKIT

🚀 Quick Start · 📋 Specifications · 📦 PyPI · 📝 Changelog

CI passing | Agent Governance Toolkit 13 members | License MIT | PyPI v4.0.0 | npm v4.0.0 | NuGet v4.0.0

openssf scorecard 7.4 | openssf best practices passing | OWASP Agentic Top 10 10/10 Covered

> **Important**
>
> **Public Preview** — production-quality, Microsoft-signed releases. May have breaking changes

https://github.com/microsoft/agent-governance-toolkit/blob/main/docs/assets/readme-banner.svg

agentexecutor.io

All Bookmarks

AX

GitHub



Declare an agentic task.
AX runs it at scale.

AX sandboxes your task, wires up its workspace, fences its network, and helps you run a large number of them per cluster. Either use a single task per agent, or compose as many as your agent needs.

Get started

View on GitHub

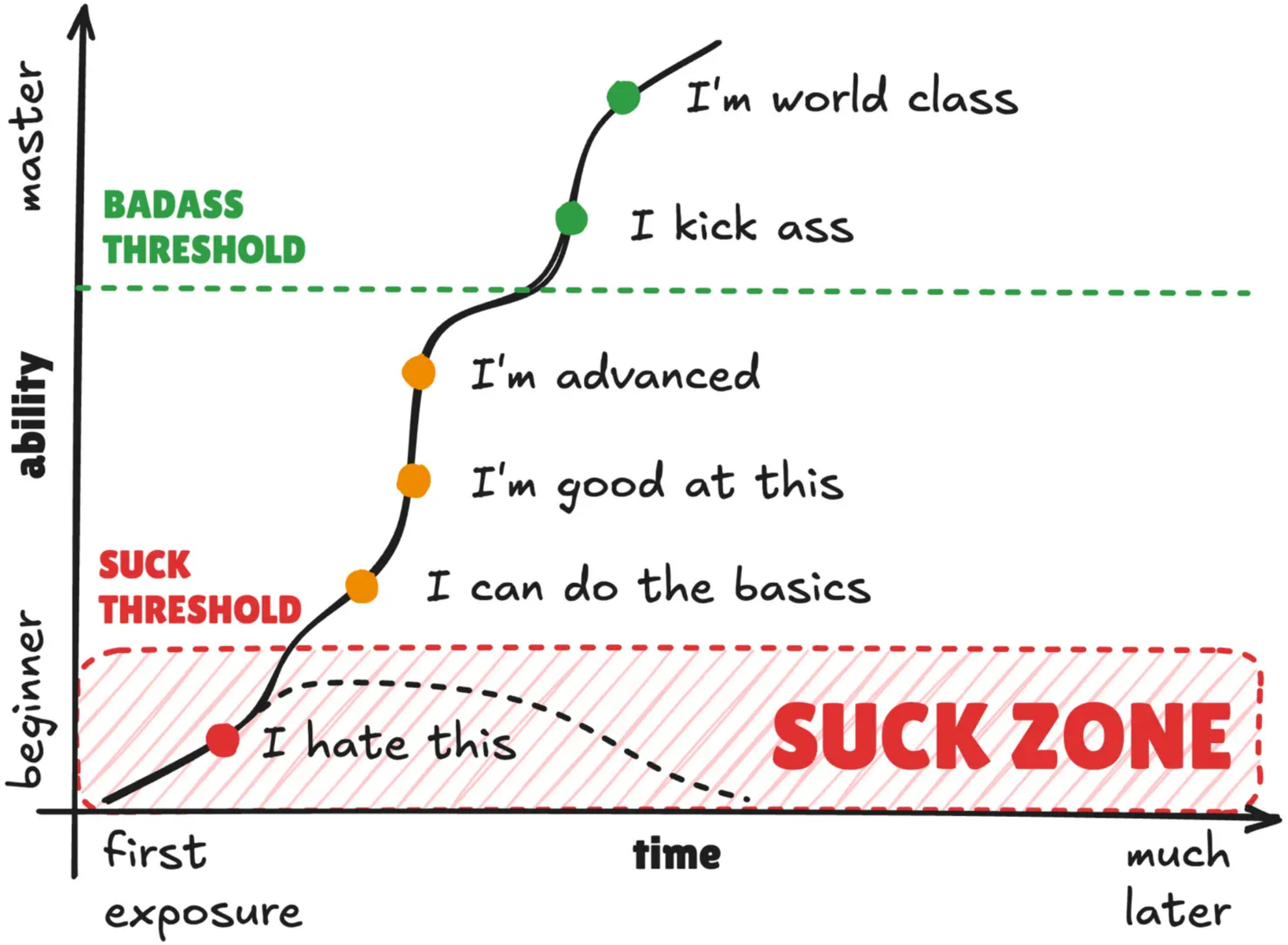
Pause

```
spec:
  workspaces:
    - name: golang
      goal: "Ensure that Go tool chain is available and is bu
  debug: true
$ ax apply -f task.yaml
workspace.ax.io/golang created
task.ax.io/test created
$ ax watch task test
Watching task default/test...
[10:42:01] Phase: Pending    Actor: test                Worke
[10:42:05] Phase: Running    Actor: test                Worke
Task reached terminal phase "Running".
$ ax get tasks
NAME   ATESPACE   PHASE     ACTOR   WORKER-IP    AGE
test   default    Running   test    10.20.3.67   5s
```

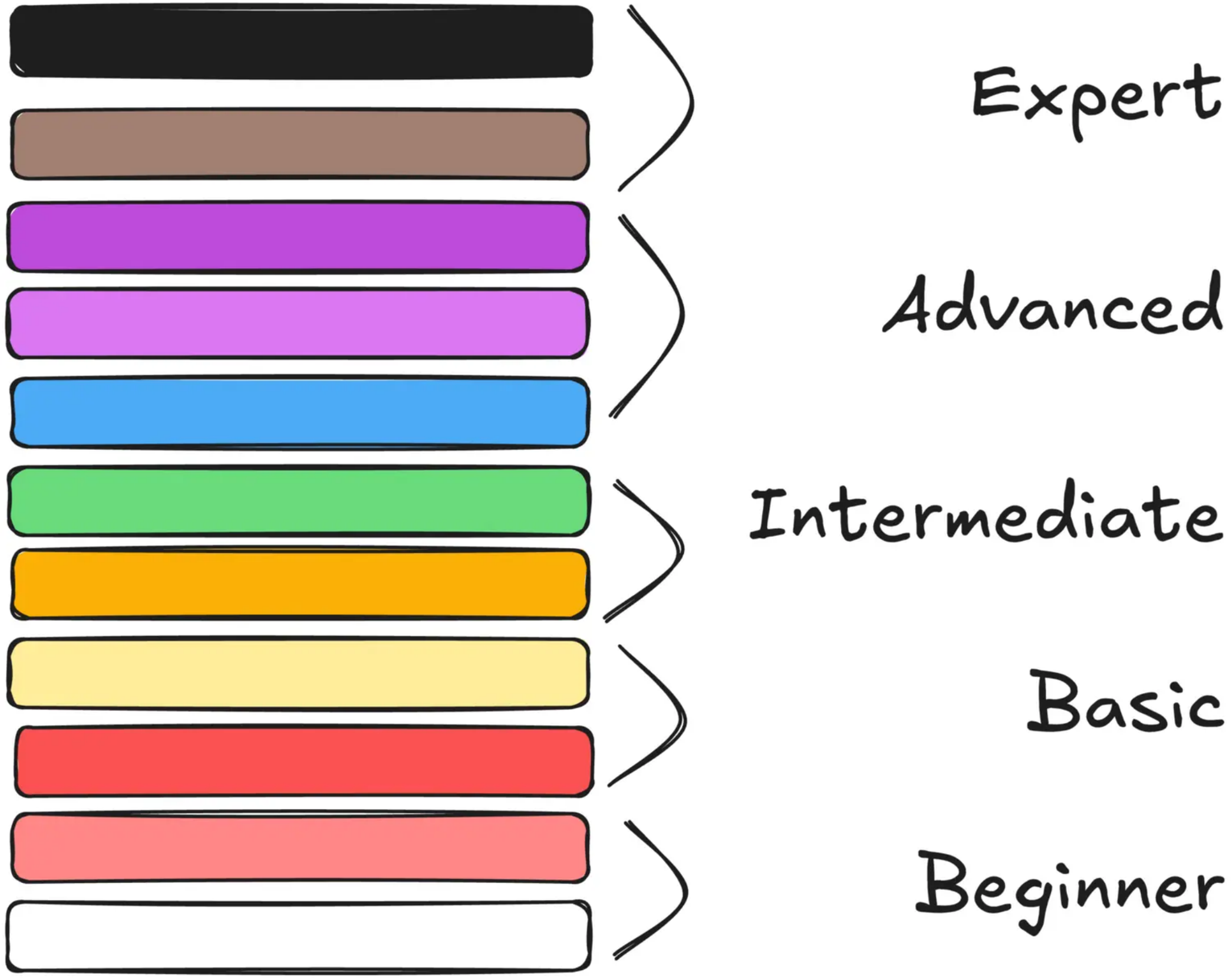
WHY AX

Agents are a new kind of workload.

They are neither microservices nor batch jobs. They accumulate state, need strict isolation, call out to model APIs and tool servers, and can waste resources if nobody is watching. AX gives you four small primitives that handle all of that declaratively.



adapted from Bad Ass by Kathy Sierra



adapted from Bad Ass by Kathy Sierra

PERFORMANCE PATH

1. Dictating
2. Commanding
3. Regulating
4. Orchestrating
5. Supervising

DICTATING

- agents do subtasks
- *short, continuous instructions*
- interactive exploration/prompt sessions
- prompts mix instructions and contextual information
- little or no project specific setup

➤COMMANDING

- pay attention to *repetitive tasks*
- start documenting your process
- start creating commands and skills
- task context files
- RPI loop

```
22
23 ## Planning process (MANDATORY)
24
25 ### Pass 1: Phase outline
26
27 Before detailed planning, produce a **phase outline only** — a numbered list of phases with:
28
29 - Phase name (the capability, or the property of the tree, the phase delivers)
30 - One-line goal
31 - Which functional gaps (from the impact analysis) it covers
32
33 **Do NOT produce detailed steps, file lists, or verification commands in pass 1.**
34
35 Present the phase outline to the user via AskUserQuestion. Wait for approval or changes before proceeding. The user may reorder, merge, split, add, or remove phases.
36
37 **Ask the open scope questions in the same call.** A question whose answer changes the step list — which phase owns a shared mechanism, what a gated instance shows,
   which of two equal quantities survives — is asked here and never assumed, because a wrong answer is re-planned and not patched.
38
39 ### Pass 2: Detailed planning
40
41 After the user approves the phase outline, produce the full implementation plan with detailed steps within each phase (see Phase Structure and Step Rules below).
42
43 ### Pass 3: Dependency analysis & parallelization map (final pass)
44
45 After the detailed plan is written, do a final pass over **all phases and steps** to derive the execution structure, and record it in the plan sidecar `roadmap/<plan>.
   plan.json`, which states the groups and the dependencies. This is the **bridge to a Workflow**: a program reads the sidecar and answers the script, so nothing is
   transcribed and nothing is re-derived. See **Dependency analysis & parallelization map (MANDATORY final pass)** below for the full spec.
46
47 ## Required sections
48
49 - Executive Summary (Current State → Target State)
50 - Architecture Overview
51 - Components Analysis
52 - Reference Files
53 - Verification cadence — which steps prove against unit tests alone, and what the barrier proves at the end
54 - Implementation Plan (vertical slices with iterative steps) — produced in pass 2
55 - Success Criteria
56 - Key Design Decisions
57
58 The execution structure is NOT a section of the document. Pass 3 writes it to the plan sidecar `roadmap/<plan>.plan.json`.
59
60 ## Never output unless explicitly requested (MANDATORY)
61
62 - Timelines
63 - Code (use pseudo-code only for non-trivial operations where clarifying complexity is crucial)
64 - Deployment sequence
65 - Benefits
66 - Rollback strategy
67
68 ## Anti-goals Guardrails (MANDATORY)
69
```

COMMANDING

- agents run tasks
- prompt files or commands, usually personal and not shared
- *mix of what to do and how*
- repetitive refactoring/correcting instructions
- rudimentary constitution
- sometimes external framework (BMAD, GSD…)

➤REGULATING

- pay attention to repetitive *parts of tasks*
- split contextual/task rules
- look for common parts of tasks
- large commands ➤ sequences
- shift-left feedback (gallery…)
- evolve your process commands to prevent repetitive problems

AUTOMATING UNPREDICTABLE

- automate *parts* that are *predictable*
- feedforward + *feedback* on unpredictable

Adriana's Workspace / My Surveys

Survey Management

Manage your surveys from this page.

+ New Survey

Draft | Open | Closed

Title	Created	Modified	Closes
Task ease: checking out	9/13/2026	9/13/2026	

Show 5 per page

REGULATING

- Plans co-authored with agents
- Humans provide refactoring and correcting instructions
- Plans are "what", common context and rules are "how"
- *Rules mostly markdown*
- Agents running on developer machines.
- Detailed code review

➤ORCHESTRATING

- pay attention to sequences of commands
- human specs ➤ iteratively refine with agents
- stabilize/generalize process files
- *common feedback ➤ computational layer*
- confidence ➤ evidence

TELEMETRY SHOWED USERS APPROVED ROUGHLY 93% OF PERMISSION PROMPTS. THE MORE APPROVALS A USER SEES, THE LESS ATTENTION THEY PAY TO EACH...

https://www.anthropic.com/engineering/how-we-contain-claude

CODIFY SECURITY RULES

```
"hooks": {
  "PreToolUse": [
  {
    "matcher": "Bash",
    "hooks": [
      {
        "type": "command",
        "command": "node $CLAUDE_PROJECT_DIR/config/regulate.js"
      }
    ]
  }],
```

PROVIDE A MEANINGFUL CORRECTION MESSAGE

```
const npxJestRule = {
    id: 'npx-jest',
    decision: 'deny',
    reason: 'Use the `jest` binary from $PATH; jest is on $PATH.',
    matchSegment: (tokens) ⇒ tokens[0] ⩵ 'npx' &&
     skipNpxFlags(tokens) ⩵ 'jest'
};
```

CODIFY RECURRING FEEDBACK

```
export const requireErrorNotify = {
 meta: {
  type: 'problem',
  docs: {
   description: 'Require workflow.error.notify() in catch blocks',
  },
  messages: {
   requireErrorNotify: `Catch blocks that set errorMessage must also call
                        workflow.error.notify() for auditing.`,
  },
 },
 create(context) {
```

...FOR EVERYTHING!

```
export const noInlineStyle = {
 meta: {
  type: 'problem',
  docs: {
   description: 'Disallow the inline `style=` HTML attribute. Move styling into a stylesheet.'
  },
  messages: {
   inlineStyle: `Inline "style=" attribute is blocked by CSP.
                 Move styling into a stylesheet — page-specific SCSS,
                 app-common ('www/styles/include/'),
                 or design system ('design/voticco/src/').`
  }
   ...
 }
};
```

PERSONAL ➤ TEAM

- common container!!!
- shared process/habitat (in version control)
- shared decision log (in version control)
- no context fragmentation (single repo or semantic search)
- *reproduce deterministic checks in CI*
- continuously focus on speeding up feedback (test/CI speed)

ORCHESTRATING

- Agents plan, humans review
- Agents run "epics"
- *Humans coordinate agents*
- semi-autonomous
- high-level code review

➤SUPERVISING

- pay attention to *variances in workflow*
- multiple evidence signals
- feedback speed/reliability
- mechanical orchestration ➤ computational layer
- observability platform

kano-top-category-close-calls
The groups of roadmap/kano-top-category-close-calls.md, with a gate between two of t…27/28 agents · 48m18s

Phases

❱ ✔	Implement A	16/16
✔	Gate A	1/1
✔	Implement B	6/6
✔	Gate B	1/1
✔	Implement C	3/3
6	Barrier	0/1
7	Check	

Implement A · 16 agents

✔	step:1.1	Opus 5.5 (1M context) · 24.4k tok	27s
✔	step:1.2	Opus 5.5 (1M context) · 30.2k tok	1m04s
✔	step:1.3	Opus 5.5 (1M context) · 31.6k tok	1m04s
✔	step:1.10	Opus 5.5 (1M context) · 43.3k tok	1m09s
✔	step:2.1	Opus 5.5 (1M context) · 28.4k tok	56s
✔	step:4.1	Opus 5.5 (1M context) · 24.4k tok	25s
✔	step:4.2	Opus 5.5 (1M context) · 49k tok	1m07s
✔	step:1.4	Opus 5.5 (1M context) · 47.6k tok	19m15s
✔	step:1.5	Opus 5.5 (1M context) · 37.6k tok	18m52s
✔	step:1.6	Opus 5.5 (1M context) · 63.5k tok	19m22s
✔	step:1.7	Opus 5.5 (1M context) · 35.9k tok	18m50s
✔	step:2.2	Opus 5.5 (1M context) · 33k tok	18m49s
✔	step:3.1	Opus 5.5 (1M context) · 28.5k tok	1m00s
✔	step:1.8	Opus 5.5 (1M context) · 65.5k tok	1m47s
✔	step:2.3	Opus 5.5 (1M context) · 45.8k tok	1m06s
✔	step:1.9	Opus 5.5 (1M context) · 42.1k tok	1m14s

↑↓ select · x stop workflow · p pause · esc back · s save

File: /builder/.claude/projects/-Users-gojkoadzic-work-votito-app/05964a44-0b3e-43ab-84dd-f4f098f6d585.jsonl




```
{"type":"mode","mode":"normal","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6d585"}
{"parentUuid":null,"isSidechain":false,"type":"user","message":{"role":"user","content":"<local-command-caveat>Caveat: The messages below were generated by the user while
{"parentUuid":"18e62bb3-0b36-4...1a-628214fedff4","isSidechain":false,"type":"user","message":{"role":"user","content":"<command-name>/clear</command-name>\n
{"parentUuid":"6f0b7dca-b714-43...fa7aaed6f430","isSidechain":false,"type":"system","subtype":"local_command","content":"<local-command-stdout></local-command-stdout
{"parentUuid":"d3407ea2-a08d-4d...0580d35fd","isSidechain":false,"promptId":"ac725141-a605-45cc-83fe-d93156803b93","type":"user","message":{"role":"user","content
{"parentUuid":"9ee2e6c4-0a48-46b...f2fb88e","isSidechain":false,"promptId":"ac725141-a605-45cc-83fe-d93156803b93","type":"user","message":{"role":"user","content
{"parentUuid":"d128c59a-84ec-4c63...","isSidechain":false,"attachment":{"type":"environment","snapshot":{"workingDirectory":"/Users/gojkoadzic/work/votito/
{"parentUuid":"52799b3c-22e5-4ff7...","isSidechain":false,"attachment":{"type":"model","identity":{"modelId":"claude-opus-5-5[1m]","marketingName":"Opus 5.
{"parentUuid":"8d488352-4d9...","attachment":{"type":"output_style_instructions","style":{"name":"Concise","prompt":"You are an i
{"parentUuid":"7c09575...tachment":{"type":"language","language":"ASD-STE100"},"type":"attachment","uuid":"3a64c508-85a
{"parentUuid":"3a64c...chment":{"type":"deferred_tools_delta","addedNames":["CronCreate","CronDelete","CronList","E
{"parentUuid":"75e...hment":{"type":"agent_listing_delta","addedTypes":["claude-code-guide","Explore","general-p
{"parentUuid":"68...ment":{"type":"mcp_instructions_delta","addedNames":["claude.ai Claude Docs"],"addedBlocks
{"parentUuid":"4a4...ment":{"type":"skill_listing","content":"- demo: Demo - Record Narrated Product Videos\n-
{"parentUuid":"d9ea...hment":{"type":"output_style","style":"Concise","turnReminder":"Be concise: lead with the r
{"parentUuid":"add1c...tachment":{"type":"command_permissions","allowedTools":[]},"type":"attachment","uuid":"47ae34c
{"type":"last-prompt",...,"leafUuid":"47ae34c4-4a31-4bfa-aba0-e2e1a27ff8b3","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6
{"type":"mode","mode":"norma...f098f6d585"}
{"type":"atis-latch","atis":"",...-84dd-f4f098f6d585"}
{"parentUuid":"47ae34c4-4a31-4bfa-aba0-e2e1a27ff8b3","isSidechain":false,"attachment":{"type":"instructions","files":[{"path":"/Users/gojkoadzic/work/votito/app/CLAUDE.md
{"parentUuid":"fb3cb99d-66bb-484f-94d2-024f129ca104","isSidechain":false,"attachment":{"type":"session_context","context":{"userEmail":"The user's email address is contac
{"parentUuid":"526159fd-27a0-4a5d-8f63-2ce1a9d5ebf0","isSidechain":false,"attachment":{"type":"date","date":"2026-09-30"},"type":"attachment","uuid":"7cecfe97-6322-4cc3-9
{"parentUuid":"7cecfe97-6322-4cc3-93c0-b7e6265fbaab","isSidechain":false,"attachment":{"type":"prompt_snapshot","systemPrompt":["\nYou are an interactive agent that helps
{"type":"last-prompt","lastPrompt":"/vtt:iterative-plan kano-top-category","leafUuid":"af702acb-3776-489d-b101-55b7c241c1ab","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6
{"type":"mode","mode":"normal","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6d585"}
{"type":"atis-latch","atis":"","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6d585"}
{"parentUuid":"af702acb-3776-489d-b101-55b7c241c1ab","isSidechain":false,"message":{"model":"claude-opus-5-5","id":"msg_011CfZevoGrmLh518GdLEwjg","type":"message","role":
{"parentUuid":"064fbddf-88ba-477c-9a4e-e71fbb63fc0b","isSidechain":false,"message":{"model":"claude-opus-5-5","id":"msg_011CfZevoGrmLh518GdLEwjg","type":"message","role":
{"parentUuid":"286694a9-2be7-4f8e-81c7-7db59f5d2250","isSidechain":false,"message":{"model":"claude-opus-5-5","id":"msg_011CfZevoGrmLh518GdLEwjg","type":"message","role":
{"parentUuid":"90e359bf-9231-4fa2-856a-6bac26b6c2d7","isSidechain":false,"attachment":{"type":"hook_success","hookName":"PreToolUse:Bash","toolUseID":"toolu_01FA1hQMwLWfv
{"parentUuid":"286694a9-2be7-4f8e-81c7-7db59f5d2250","isSidechain":false,"promptId":"ac725141-a605-45cc-83fe-d93156803b93","type":"user","message":{"role":"user","content
{"parentUuid":"0a157eec-a03a-4644-a537-5eaaa6540a8e","isSidechain":false,"attachment":{"type":"hook_success","hookName":"PreToolUse:Bash","toolUseID":"toolu_01UeafHK99ZrP
{"parentUuid":"90e359bf-9231-4fa2-856a-6bac26b6c2d7","isSidechain":false,"promptId":"ac725141-a605-45cc-83fe-d93156803b93","type":"user","message":{"role":"user","content
{"parentUuid":"d4af0238-74c6-424c-9d01-76757acf567a","isSidechain":false,"attachment":{"type":"prompt_snapshot","systemPrompt":["\nYou are an interactive agent that helps
{"parentUuid":"5308bc6b-751f-497f-976b-52287ea9e3bb","isSidechain":false,"attachment":{"type":"deferred_tools_record","entries":[{"name":"mcp__claude_ai_Claude_Docs__batc
{"parentUuid":"bc98bff0-44af-4bab-b4e3-02d456d84700","isSidechain":false,"attachment":{"type":"total_tokens_reminder","text":"<total_tokens>14784618 tokens left</total_to
{"parentUuid":"01547fa4-bd64-4866-91db-12bb3239ad83","isSidechain":false,"attachment":{"type":"output_style","style":"Concise","turnReminder":"Be concise: lead with the r
{"type":"last-prompt","lastPrompt":"/vtt:iterative-plan kano-top-category","leafUuid":"0f62c7ab-fadf-4e7f-aa74-374cea5dd714","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6
{"type":"mode","mode":"normal","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6d585"}
{"type":"atis-latch","atis":"","sessionId":"05964a44-0b3e-43ab-84dd-f4f098f6d585"}
{"parentUuid":"0f62c7ab-fadf-4e7f-aa74-374cea5dd714","isSidechain":false,"message":{"model":"claude-opus-5-5","id":"msg_011CfZewzKAfUxA8Avpmy9cP","type":"message","role":
{"parentUuid":"f33a95bf-a483-40ce-b245-2855114d0f28","isSidechain":false,"promptId":"ac725141-a605-45cc-83fe-d93156803b93","type":"user","message":{"role":"user","content
{"parentUuid":"67e7d023-495f-498d-a419-661dc3deb912","isSidechain":false,"attachment":{"type":"total_tokens_reminder","text":"<total_tokens>14784378 tokens left</total_to
```

`eslint data/biketheft/src/run-gazetteer-stage.ts data/biketheft/tests/run-gazetteer-stage.test.ts`

Step 1.3: The entry reads the share and the path of the record

Goal: the recipe states the share and the third path, and neither reaches the key of the node.

Files to create: `none`

Files to modify:

1. `data/biketheft/src/gazetteer-stage-entry.ts`
2. `data/biketheft/tests/gazetteer-stage-entry.test.ts`

Files to delete: `none`

Written also by: `none`

The entry reads `--overlap-share` and `--overlaps` off the command line. **The share reaches the stage and it reaches NO key**, so two shares answer one table and a second reading makes nothing again. The `panelWidestMetres` setting of the key stays as it is, beside the gazetteer word, the join distance and the floor share.

A run that states no path for the record is refused by name, as a run that states no path for the inventory is. A run that states no share takes the default the entry holds, which is half of the smaller group.

Verification:
`jest data/biketheft/tests/gazetteer-stage-entry.test.ts`, then
`eslint data/biketheft/src/gazetteer-stage-entry.ts data/biketheft/tests/gazetteer-stage-entry.test.ts`

Step 1.4: The node declares three outputs

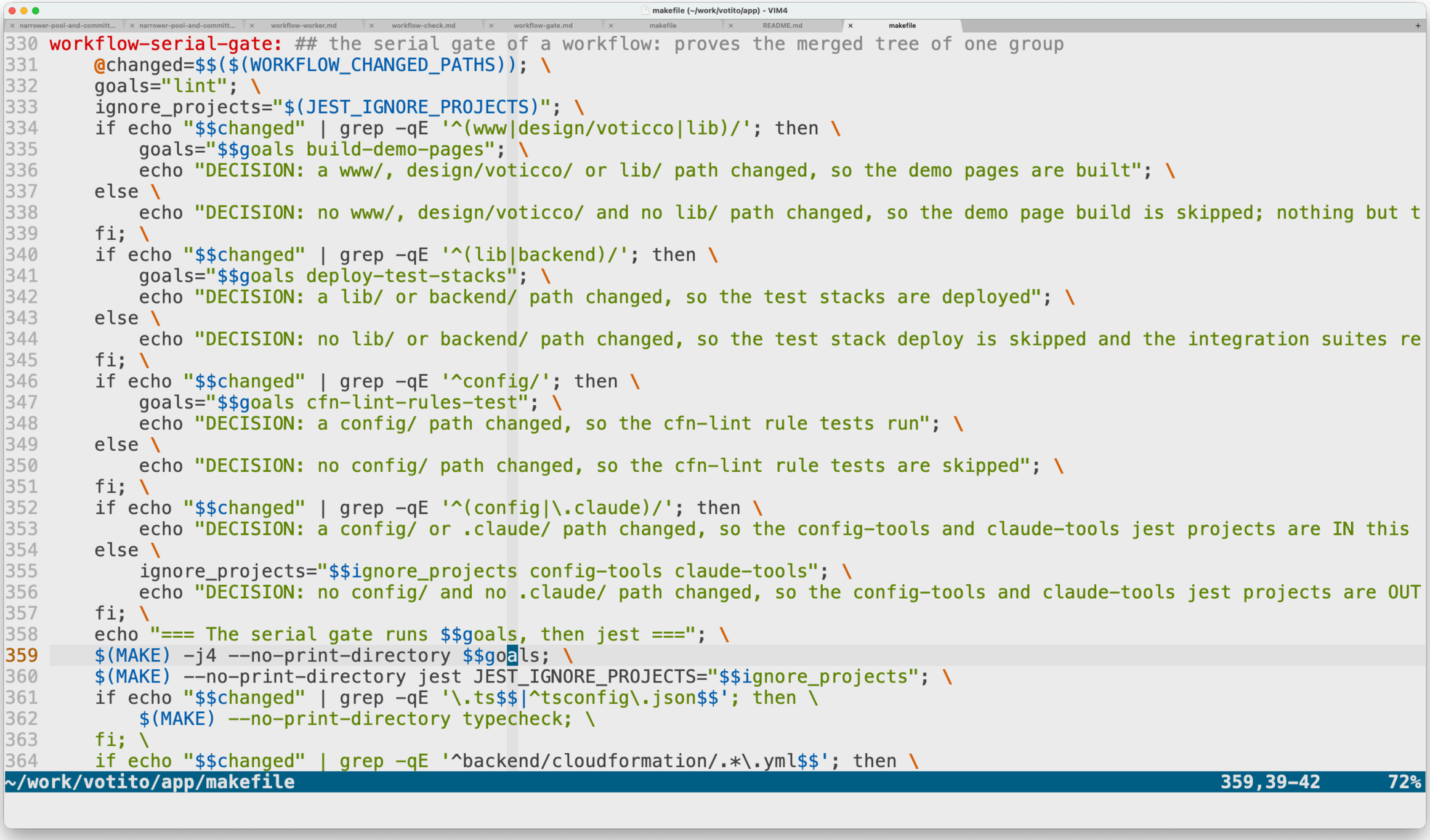
Goal: the rule of the gazetteer node names the third file, and the pipeline

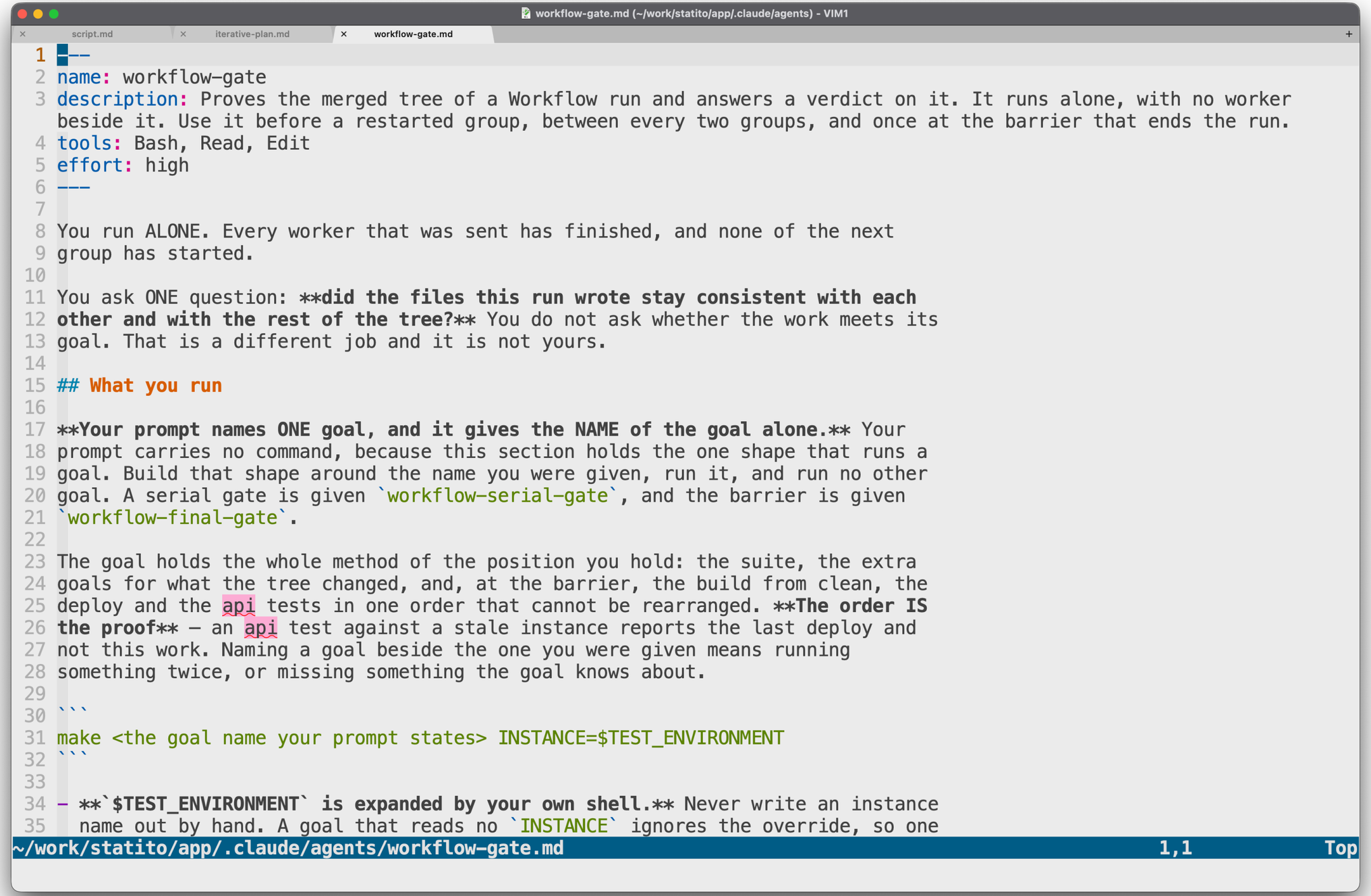
narrower-pool-and-committed-set.plan.json (~/work/statito/app/roadmap) - VIM4

narrower-pool-and-committ... | narrower-pool-and-committ... | workflow-worker.md | workflow-check.md | workflow-gate.md | makefile | README.md | makefile

```
{
  "groups": [
    { "name": "A", "steps": ["1.1", "1.2", "1.3", "1.4"] },
    { "name": "B", "steps": ["2.1", "2.2", "2.3"] },
    { "name": "C", "steps": ["3.1", "3.2", "3.3", "4.1", "4.2"] },
    { "name": "D", "steps": ["5.1", "5.2"] }
  ],
  "dependencies": {
    "1.2": ["1.1"],
    "1.3": ["1.2"],
    "1.4": ["1.3"],
    "2.2": ["2.1"],
    "2.3": ["2.1"],
    "3.2": ["3.1"],
    "3.3": ["3.2", "2.3"],
    "4.1": ["3.2"],
    "4.2": ["4.1"],
    "5.1": ["4.2", "3.3"],
    "5.2": ["4.2"]
  }
}
```

roadmap/narrower-pool-and-committed-set.plan.json 1,1 All


makefile (~/work/votito/app) - VIM4
narrower-pool-and-committ...
narrower-pool-and-committ...
workflow-worker.md
workflow-check.md
workflow-gate.md
makefile
README.md
makefile
workflow-serial-gate: ## the serial gate of a workflow: proves the merged tree of one group
@changed=$$($(WORKFLOW_CHANGED_PATHS)); \
goals="lint"; \
ignore_projects="$(JEST_IGNORE_PROJECTS)"; \
if echo "$$changed" | grep -qE '^(www|design/voticco|lib)/'; then \
goals="$$goals build-demo-pages"; \
echo "DECISION: a www/, design/voticco/ or lib/ path changed, so the demo pages are built"; \
else \
echo "DECISION: no www/, design/voticco/ and no lib/ path changed, so the demo page build is skipped; nothing but t
fi; \
if echo "$$changed" | grep -qE '^(lib|backend)/'; then \
goals="$$goals deploy-test-stacks"; \
echo "DECISION: a lib/ or backend/ path changed, so the test stacks are deployed"; \
else \
echo "DECISION: no lib/ or backend/ path changed, so the test stack deploy is skipped and the integration suites re
fi; \
if echo "$$changed" | grep -qE '^config/'; then \
goals="$$goals cfn-lint-rules-test"; \
echo "DECISION: a config/ path changed, so the cfn-lint rule tests run"; \
else \
echo "DECISION: no config/ path changed, so the cfn-lint rule tests are skipped"; \
fi; \
if echo "$$changed" | grep -qE '^(config|\.claude)/'; then \
echo "DECISION: a config/ or .claude/ path changed, so the config-tools and claude-tools jest projects are IN this
else \
ignore_projects="$$ignore_projects config-tools claude-tools"; \
echo "DECISION: no config/ and no .claude/ path changed, so the config-tools and claude-tools jest projects are OUT
fi; \
echo "=== The serial gate runs $$goals, then jest ==="; \
$(MAKE) -j4 --no-print-directory $$goals; \
$(MAKE) --no-print-directory jest JEST_IGNORE_PROJECTS="$$ignore_projects"; \
if echo "$$changed" | grep -qE '\.ts$$|^tsconfig\.json$$'; then \
$(MAKE) --no-print-directory typecheck; \
fi; \
if echo "$$changed" | grep -qE '^backend/cloudformation/.*\.yml$$'; then \
~/work/votito/app/makefile
359,39-42
72%


workflow-gate.md (~/work/statito/app/.claude/agents) - VIM1
script.md
iterative-plan.md
workflow-gate.md
---
name: workflow-gate
description: Proves the merged tree of a Workflow run and answers a verdict on it. It runs alone, with no worker beside it. Use it before a restarted group, between every two groups, and once at the barrier that ends the run.
tools: Bash, Read, Edit
effort: high

You run ALONE. Every worker that was sent has finished, and none of the next group has started.
You ask ONE question: **did the files this run wrote stay consistent with each other and with the rest of the tree?** You do not ask whether the work meets its goal. That is a different job and it is not yours.
What you run
Your prompt names ONE goal, and it gives the NAME of the goal alone. Your prompt carries no command, because this section holds the one shape that runs a goal. Build that shape around the name you were given, run it, and run no other goal. A serial gate is given `workflow-serial-gate`, and the barrier is given `workflow-final-gate`.
The goal holds the whole method of the position you hold: the suite, the extra goals for what the tree changed, and, at the barrier, the build from clean, the deploy and the api tests in one order that cannot be rearranged. **The order IS the proof** — an api test against a stale instance reports the last deploy and not this work. Naming a goal beside the one you were given means running something twice, or missing something the goal knows about.
```
make <the goal name your prompt states> INSTANCE=$TEST_ENVIRONMENT
```
- **`$TEST_ENVIRONMENT` is expanded by your own shell.** Never write an instance name out by hand. A goal that reads no `INSTANCE` ignores the override, so one
~/work/statito/app/.claude/agents/workflow-gate.md 1,1 Top

WORKFLOW: INFERENCE ➤ COMPUTATIONAL

- 80% token reduction
- 6-10 times faster
- no testing gaps
- no approval requests

SUPERVISING

- agents plan, humans review
- iteratively refine spec, multiple feedback levels
- agents provide comprehensive evidence
- humans *certify evidence*
- humans *drive platform optimization*
- agents potentially running in a central runtime

THROUGH FEEDBACK... A MECHANISM CAN EMBODY PURPOSE

Mitchell Waldrop, The Dream Machine

META-AUTOMATION: AUTOMATE THE AUTOMATION

- enable agents to *self-evaluate* and *self-correct*
- design "*fitness*" functions, much more detailed than with humans
- ask agents to *build custom tools*/rules to check
- make sure *recovery* instructions are in the *error messages*

USE AI ADOPTION AS AN EXCUSE TO TIGHTEN UP YOUR PROCESS

MORE INFO ➤ GOJKO@VOTITO.COM