CONTINUOUS AGENTIC DEPLOYMENT

Nobody touched the keyboard.

A request became code — tested, reviewed, and shipped to production. Built by AI agents.

Generating code is the easy 10%.

The other 90% — structure, retries, validation, cost control, and a human staying in charge — is a CI/CD problem, not a prompting one.

```
CI / CD  →  CAD
```

Eight stations. One piece of work.

CAD — the engine

green = agent · orange = human gate · blue = deterministic · grey = column

SA/BA = issue analysis · Dev = code generation · Test = testing · Deploy = deployment

CAD = the engine (the dashed box) · CAD Console = the app it builds

GitHub is the runtime.

Projects
the state machine

Actions
the agents

Issue comments
the context bus

No orchestrator. No database. The board is the DB.

Every station is the same shape.

```
a workflow        .github/workflows/agent-<role>.yml
a system prompt   prompts/<role>-system.md
a WIP limit       vars.WIP_<ROLE>
a model           vars.MODEL_<ROLE>
a board column    the Project status
```

The dispatcher runs the shape. The engine doesn't change.

Prompts, not code.

Each agent's job lives in a markdown file. Swap the file, swap the agent.

```
prompts/saba-system.md
prompts/dev-system.md
prompts/test-system.md
```

Add a station in four steps.

1
drop a prompt

2
copy a workflow

3
add a WIP var

4
add a column

The engine barely changed — one line. That's inversion of control — that's the framework.

Trust the harness, not the model.

Retry
self-correct, capped 3×

Eval gate
validate before promote

Isolation
a branch per card

Agents are non-deterministic. The harness around them is not.

Agents earn the deploy. A human says go.

agentic
analyze · code · test · resolve

human
approve

deterministic
merge · build · ship

Stochastic where it's cheap to be wrong. Deterministic where it isn't. No model runs docker push.

See the line — and what it cost.

$0.67	$8.20	9	pinned
median $/feature	total spend	features shipped	deploy version

The board is the live observability layer — the dashboard is the pipeline watching itself. Cost, throughput, and the deployed version, read off the line.

Every failure had a fallback.

Two widgets edit one file → resolver agent rebases + auto-redeploys

A plausible-but-wrong change → the eval gate refuses to promote

A stage stalls or errors → item stays put, alert fires, re-dispatch resumes

An ambiguous request → the SA/BA gate: the spec is the contract

The bottleneck is good issues and an additive codebase — not a bigger model.

Six reasons it's a framework.

1. Pull-based dispatcher — inversion of control
2. A uniform station contract
3. An externalized prompt library
4. GitHub Projects as the state machine
5. Convention-driven reliability primitives
6. Per-station model + WIP as config

One honest gap: the engine still ships fused to one app. Extracting it is in the repo.

Clone it.

```
git clone github.com/errmakov/cad
# set vars · seed a backlog
gh workflow run dispatcher.yml
```

Engine, workflows, prompts, cost scripts — all open source.

CI/CD — with agents in the loop, and a human on the gate.

github.com/errmakov/cad · the framework

cadconsole.errmakov.com · the app it builds

Thanks for watching.