upsun

Formerly Platform.sh

Making Coding Agents Reliable

Spec-Driven Development &
The 8 Pillars of Verification

Hey. I'm G/

Love to tinker with stuff.
Code, cameras, watches.

Love buying domain names for
unfinished side projects & setting up infrastructure
and pipelines.

Hate boring repetitive tasks.

Started coding in 98.
Started vibe-coding in 2023.
Started spec-coding in 2025.
...

The Promise

Point an agent at a ticket...

...come back to a merged PR.

But here's what actually happens

issues and unknowns everywhere.

The real bottleneck isn't the model

It's your ability to both have a quality **input**
& verify the **output**.

Without intent & verification,
AI-generated code is just...

Text.

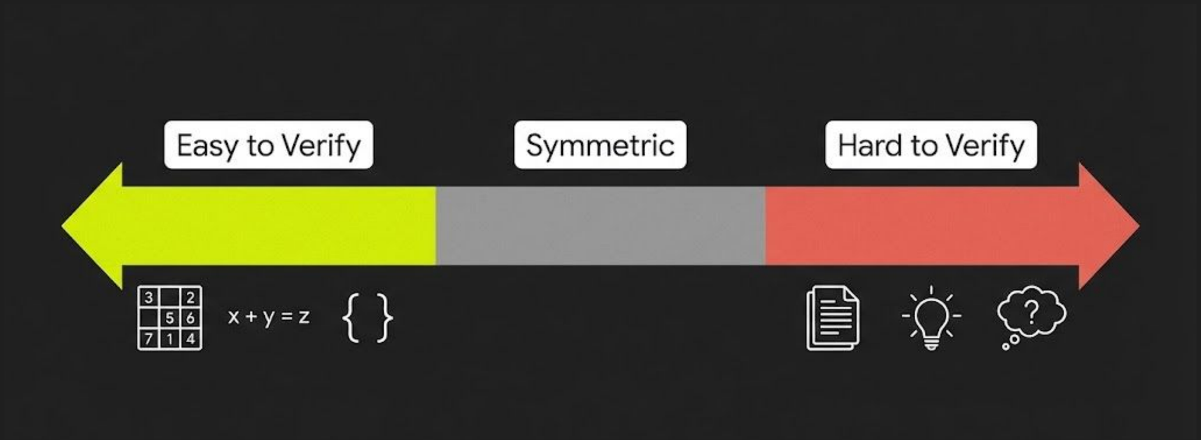
Text that might break production.

"

Software 1.0 easily automates what you can specify.
Software 2.0 easily automates what you can verify.

— Andrej Karpathy

The Asymmetry of Verification



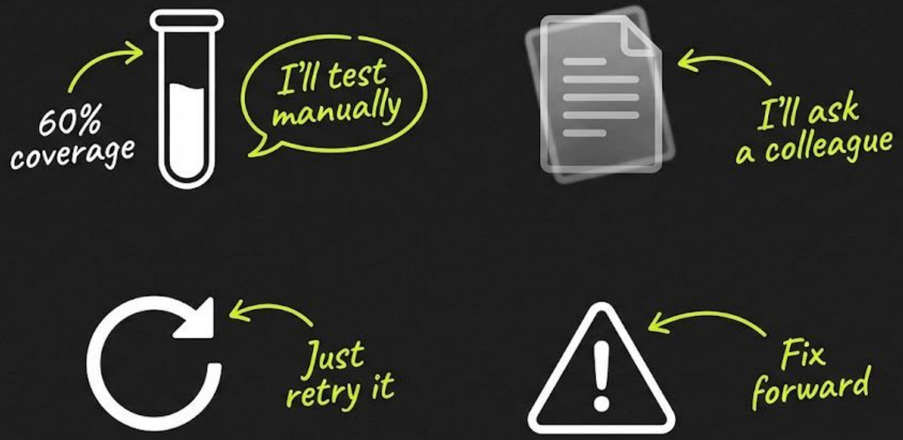
Checking is far easier than generating.

Code is highly verifiable

Tests. Types. Lints. Deployments.

When your pipelines & infrastructure support it.

Why humans cope but agents crash



Humans
can work around
incomplete
infrastructure

AI agents can't do this

No institutional knowledge.
No intuition.
No context.

What breaks agents

Missing	Result
No tests	Can't validate correctness
No specs/docs	Wrong assumptions
Flaky builds	Can't distinguish bugs from code/infra
No observability	Can't infer the result of actual deployment
No preview environments	Ships blind

The result?

”AI Slop”

Code that looks plausible but subtly degrades your codebase.

upsun

Formerly Platform.sh

Spec-Driven Development

Lock intent before implementation

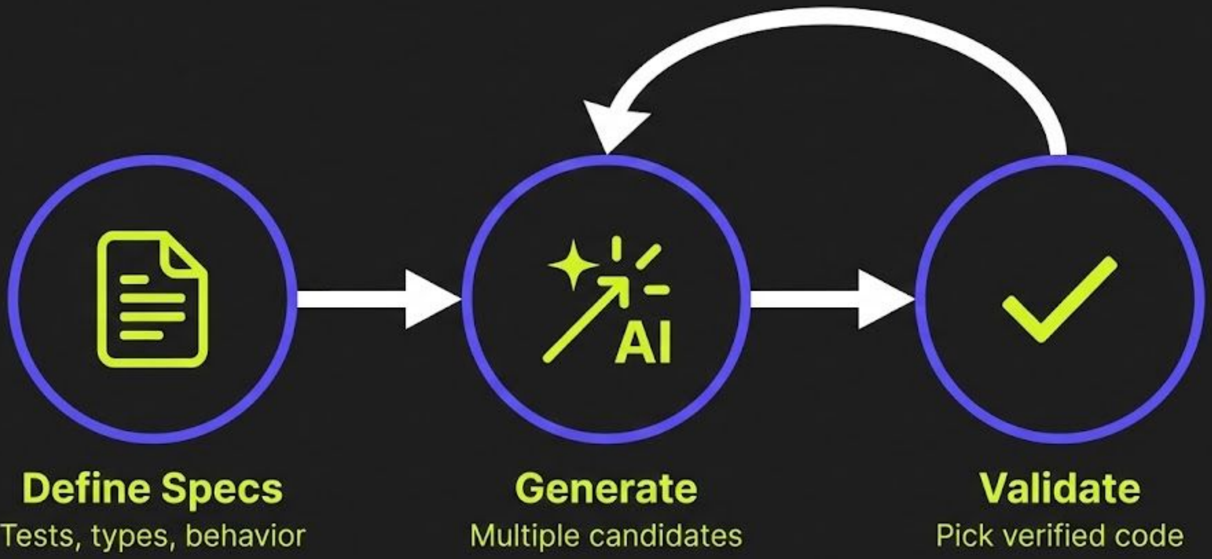
Prompt → Generate → Hope / Ragequit

Unreliable. Hard to debug. Doesn't scale.

Specs + Tests → Generate → Validate → Iterate

Reliable. Debuggable. Scales to complex tasks.

The Three-Step Process



Why this works

"

With strong verification, you can search for solutions instead of crafting them by hand.

Specs lock intent — no ambiguity

Changes are reviewable — transparent scope

Clear success criteria — tests pass or they don't

Debugging is tractable — know what should happen

upsun

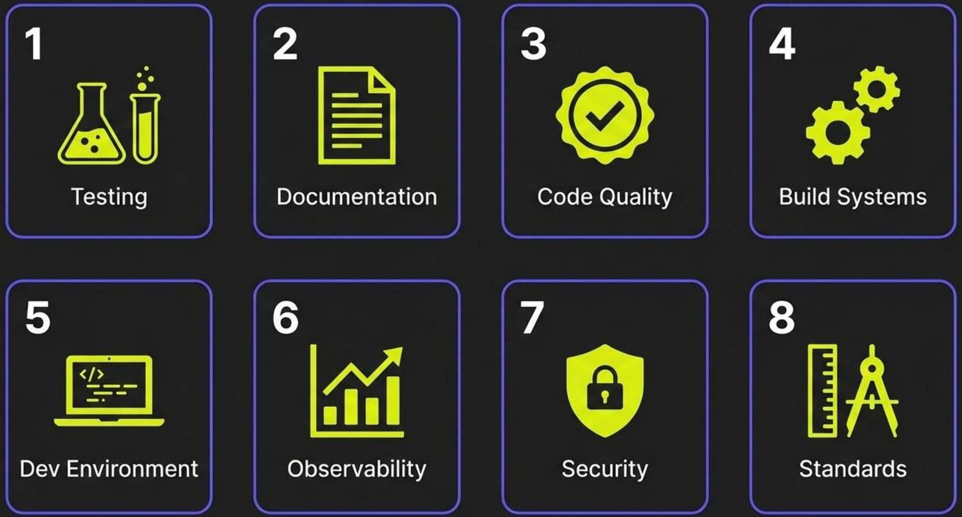
Formerly Platform.sh

The 8 Pillars of Verification

The infrastructure that makes agents reliable

Getting your codebase agent-ready

Gaps in any pillar limit agent autonomy.



1 Testing

Without testing, agent can't validate its changes work.

Generates plausible-looking code that breaks in production.

What agents need from testing:

High coverage — more tests = more signals

Fast execution — agents iterate rapidly

Deterministic results — flaky tests are worse than no tests

2

Documentation / Specs

Agent makes wrong assumptions. Break contracts.

Codebase becomes harder to maintain.

What agents need from documentation:

Up-to-date — stale docs = wrong assumptions

System behavior — how components interact

Integration points — where code touches other systems

Known limitations — edge cases, gotchas, history

3 Code quality

The Standards Enforcer

Agent generates inconsistent code.

Codebase becomes harder to maintain.

What agents need from documentation:

Strict enforcement — linter fails = agent retries

Automated gates — not suggestions, requirements

Clear error messages — what violated the rule

4 Build Systems

The Reproducibility Layer

Agent can't distinguish its bugs from infrastructure.
Wastes cycles on phantom errors.

What agents need from build systems:

Reproducible — same commit = same environment
Fast failure — errors surface immediately
No mystery failures — "works on my machine" is invisible to agents

5 Preview environments

Agent has no way to confirm before production.

Ships blind or requires human verification.

What agents need from build preview environments:

Production parity — test env must match prod (incl. data)
Fast provisioning — minutes, not hours
Isolation — each experiment runs independently

6 Observability

The Feedback Signal

Agent has no way to confirm before production.

Ships blind or requires human verification.

What agents need from observability:

Structured logs — JSON, not unstructured dumps
Clear metrics — baselines, error rates, resources
Actionable traces & Performance profiles — connect errors to code paths

7 Security

Agent introduces vulnerabilities.

Issues accumulate undetected. Breach risk increases.

What agents need from security:

Automated scanning — every change, every time
Dependency audit — every package, recurring
Clear reports — what's wrong, how to fix it
Secrets management — no hardcoded credentials

8 Standards

Agent generates inconsistent code.

Codebase becomes unpredictable. Debt is increased.

What agents need from standards:

Documented conventions — written, not tribal knowledge
Tooling enforcement — automation, not code review
Pattern libraries — examples of how things should be done

The 8 Pillars: Summary

Pillar	Core Question	How Upsun helps
Testing	Does it work?	E2E Testing compatible
Documentation	What should it do?	-
Code Quality	Does it meet standards?	Bundled linters in the CI
Build Systems	Can it compile reliably?	We build and deploy
Dev Environment	Can it test safely?	On the fly clone env.
Observability	What happened?	Profiling & APM & Tests
Security	Is it safe?	Sandboxed environments
Standards	Is it consistent?	-

Quick mental exercise

Which pillar is your biggest gap right now?

That's where your agents will struggle most.

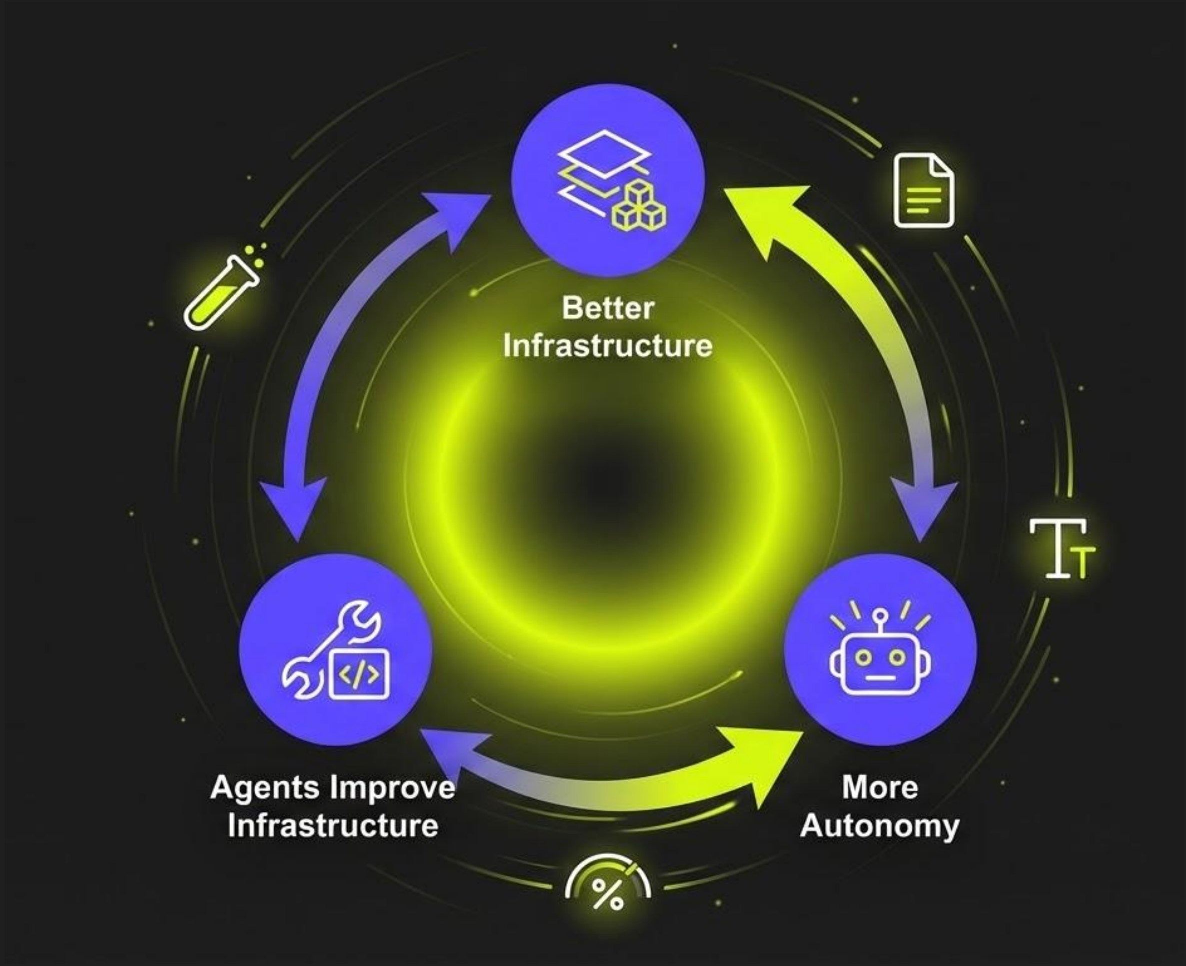
Which pillar is your biggest gap right now?

Infrastructure Checklist

Assess yourself on 80 criterias.

https://gist.github.com/gmoigneu/a963b595ac238ad2d2260ebb8b29f048

The Flywheel



Tests. Specs & Docs. Types. Coverage.

The loop accelerates.

"

*Build the verification layer,
and the autonomy follows.*

Resources

Karpathy — "Verifiability" (karpathy.bearblog.dev)

Jason Wei — "Asymmetry of Verification"

OpenSpec — github.com/openspec/openspec

upsun
Formerly Platform.sh

Thank you.

Guillaume Moigneu
Field CTO, Upsun
guillaume@upsun.com

LinkedIn