P0 INCIDENT · ON-CALL: YOU

Your Agent Built It. Who the Runs It?

Soon, you'll be on call for code you never read.

Your job is no longer to know every line.

It's to manage the system that runs it.

Ran Tavory

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA
Predictive assistance

AI IDE ERA
Conversational editing

AGENT ERA
Autonomous execution

MULTI-AGENT ERA
Orchestrated systems

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA
Predictive assistance

AI IDE ERA
Conversational editing

AGENT ERA
Autonomous execution

MULTI-AGENT ERA
Orchestrated systems

2021

GitHub Copilot

Single functions
Boilerplate
~10–50 LOC

sec – min

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA	AI IDE ERA	AGENT ERA	MULTI-AGENT ERA
Predictive assistance	Conversational editing	Autonomous execution	Orchestrated systems

2021

GitHub Copilot

Single functions
Boilerplate
~10–50 LOC

sec – min

2023

Cursor

Multi-file edits
Refactors
~100–500 LOC

minutes

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA	AI IDE ERA	AGENT ERA	MULTI-AGENT ERA
Predictive assistance	Conversational editing	Autonomous execution	Orchestrated systems

2021

GitHub Copilot

Single functions
Boilerplate
~10–50 LOC

sec – min

2023

Cursor

Multi-file edits
Refactors
~100–500 LOC

minutes

aider

2023

Aider

Repo patching
Iterative debug
Git-aware CLI

tens of min

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA	AI IDE ERA	AGENT ERA	MULTI-AGENT ERA
Predictive assistance	Conversational editing	Autonomous execution	Orchestrated systems

2021

GitHub Copilot

Single functions
Boilerplate
~10–50 LOC

sec – min

2023

Cursor

Multi-file edits
Refactors
~100–500 LOC

minutes

aider

2023

Aider

Repo patching
Iterative debug
Git-aware CLI

tens of min

2024

Windsurf

Autonomous nav
Cross-repo
Hour-scale

multi-hour

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA	AI IDE ERA	AGENT ERA	MULTI-AGENT ERA
Predictive assistance	Conversational editing	Autonomous execution	Orchestrated systems

2021
GitHub Copilot
Single functions
Boilerplate
~10–50 LOC

sec – min

2023
Cursor
Multi-file edits
Refactors
~100–500 LOC

minutes

aider

2023
Aider
Repo patching
Iterative debug
Git-aware CLI

tens of min

2024
Windsurf
Autonomous nav
Cross-repo
Hour-scale

multi-hour

2025
Claude Code
Repo-wide flows
Large refactors
Day-scale exec

hours – days

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA Predictive assistance	AI IDE ERA Conversational editing	AGENT ERA Autonomous execution	MULTI-AGENT ERA Orchestrated systems

2021
GitHub Copilot
Single functions
Boilerplate
~10–50 LOC

sec – min

2023
Cursor
Multi-file edits
Refactors
~100–500 LOC

minutes

aider

2023
Aider
Repo patching
Iterative debug
Git-aware CLI

tens of min

2024
Windsurf
Autonomous nav
Cross-repo
Hour-scale

multi-hour

2025
Claude Code
Repo-wide flows
Large refactors
Day-scale exec

hours – days

2026
Codex
Long-running tasks
Planning + exec
Tool orchestration

multi-day

The Evolution of Coding Agents 2021 → 2026

AUTOCOMPLETE ERA Predictive assistance	AI IDE ERA Conversational editing	AGENT ERA Autonomous execution	MULTI-AGENT ERA Orchestrated systems

2021
GitHub Copilot
Single functions
Boilerplate
~10–50 LOC

sec – min

2023
Cursor
Multi-file edits
Refactors
~100–500 LOC

minutes

aider

2023
Aider
Repo patching
Iterative debug
Git-aware CLI

tens of min

2024
Windsurf
Autonomous nav
Cross-repo
Hour-scale

multi-hour

2025
Claude Code
Repo-wide flows
Large refactors
Day-scale exec

hours – days

2026
Codex
Long-running tasks
Planning + exec
Tool orchestration

multi-day

2026
Cursor 3
Coordinated agents
Persistent memory
Multi-day projects

multi-day

The Evolution of Coding Agents 2021 → 2026

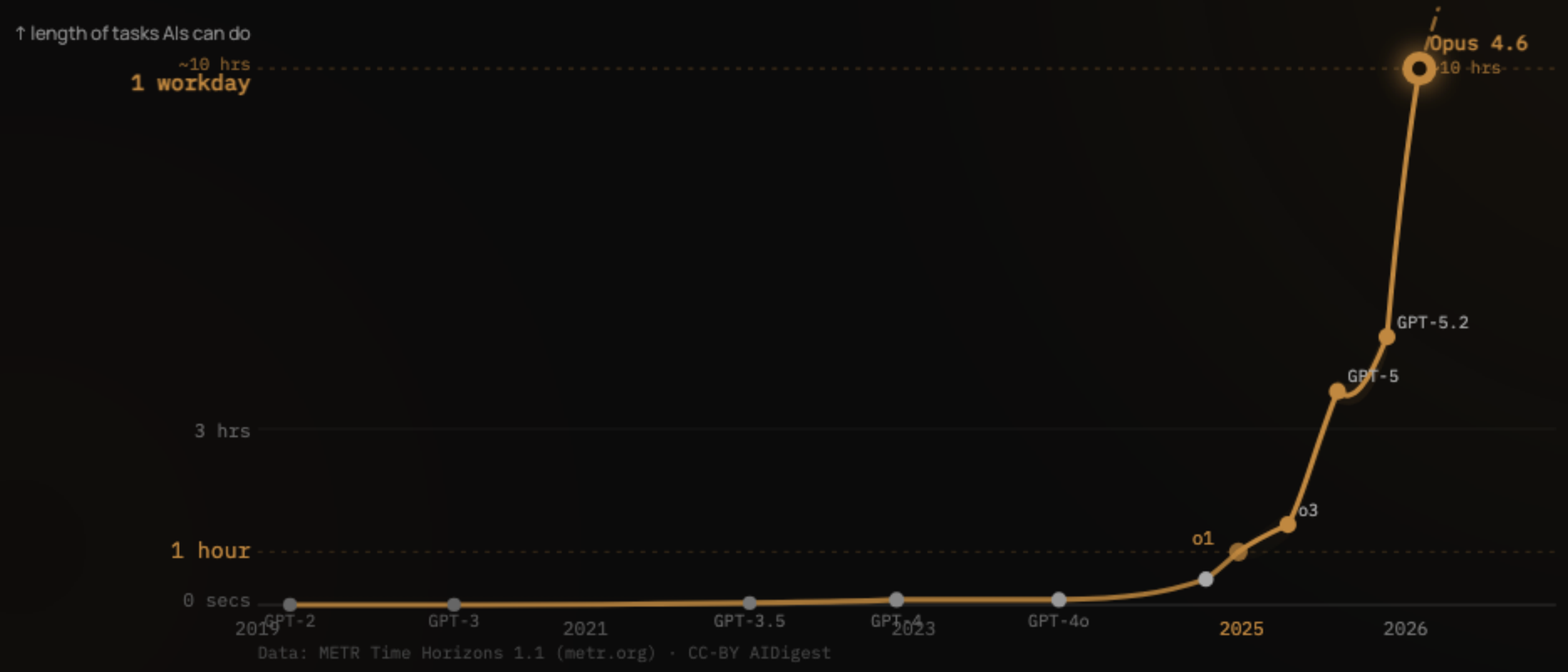
AUTOCOMPLETE ERA Predictive assistance	AI IDE ERA Conversational editing	AGENT ERA Autonomous execution	MULTI-AGENT ERA Orchestrated systems

2021	2023	2023	2024	2025	2026	2026
GitHub Copilot	**Cursor**	**Aider**	**Windsurf**	**Claude Code**	**Codex**	**Cursor 3**
Single functions Boilerplate ~10–50 LOC	Multi-file edits Refactors ~100–500 LOC	Repo patching Iterative debug Git-aware CLI	Autonomous nav Cross-repo Hour-scale	Repo-wide flows Large refactors Day-scale exec	Long-running tasks Planning + exec Tool orchestration	Coordinated agents Persistent memory Multi-day projects
sec – min	minutes	tens of min	multi-hour	hours – days	multi-day	multi-day

TASK HORIZON GROWTH

30 sec	5 min	1 hour	1 workday	multi-day
token prediction	conv. editing	agentic loops	repo-wide tasks	supervised execution

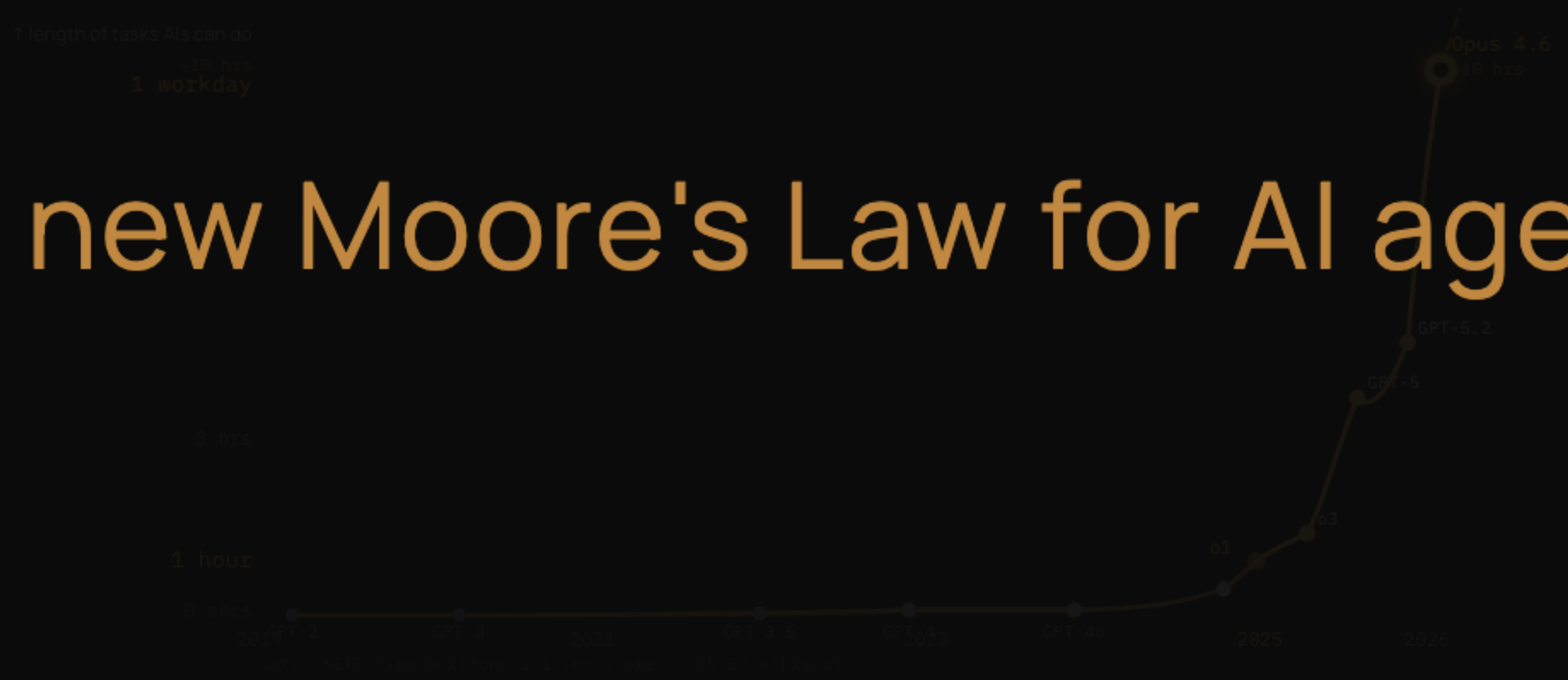
Coding agents, by how long a task they can complete.



Source: METR time-horizons study — exponential growth, no sign of slowing.

THE LONGER THE TASK, THE LOWER YOUR INTIMACY WITH THE OUTPUT.

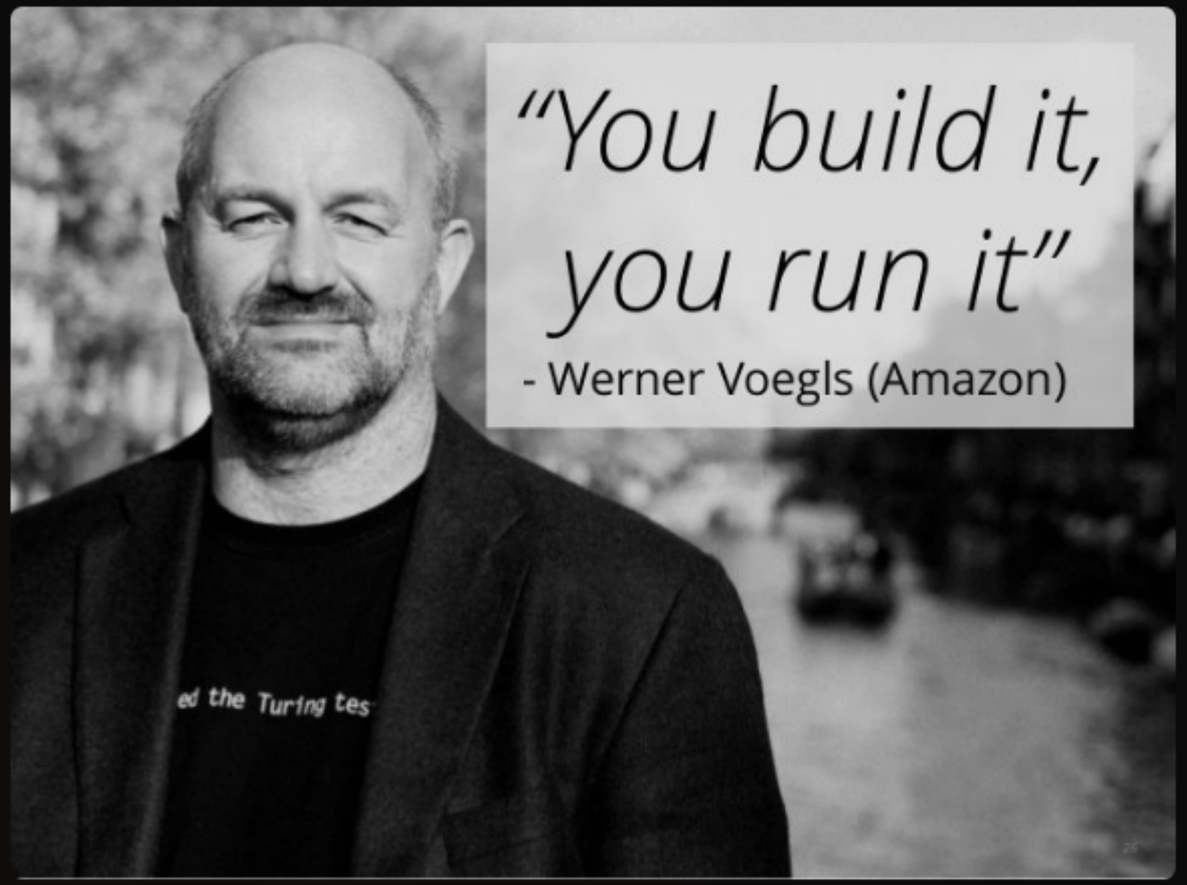
Coding agents, by how long a task they can complete.



A new Moore's Law for AI agents.

Source: METR time-horizons study — exponential growth, no sign of slowing.

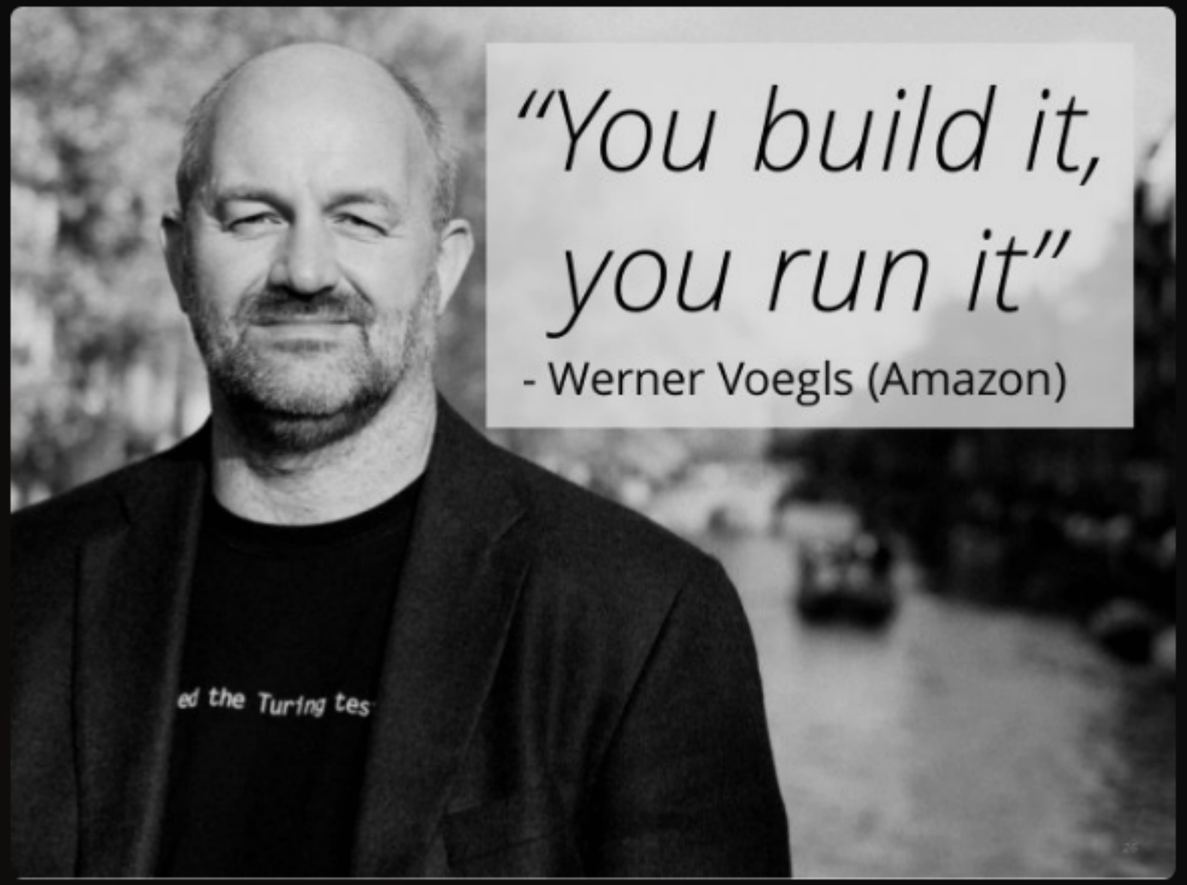
THE LONGER THE TASK, THE LOWER YOUR INTIMACY WITH THE OUTPUT.



WERNER VOGELS · AMAZON · 2006

- Ownership and quality
- Reduced friction
- Cultural shift

WERNER VOGELS · AMAZON · 2006



You didn't build it.

Your agent built it.

So who the 🤬 runs it?

Three forces making production harder.

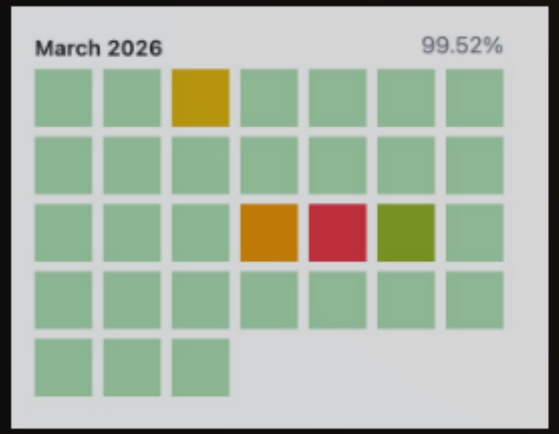
Force 1 — Volume

Force 2 — Breadth

Force 3 — Lost intimacy

More code. More surface area for failure.

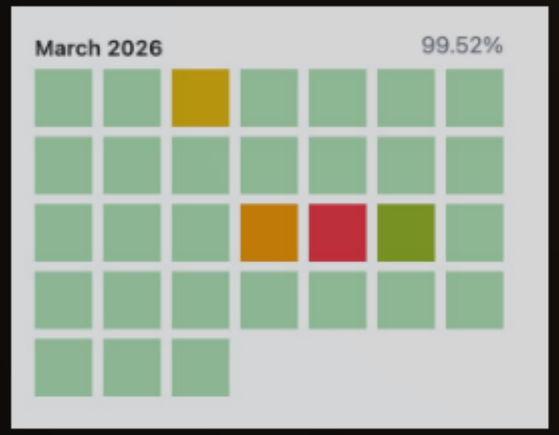
- **1 billion commits** on GitHub in 2025 — 25% YoY growth
- **26.9%** of production code is AI-authored (early 2026; n=4.2M devs)
- Honest range: **27–42%** depending on "generated" vs "assisted"
- Quality is following volume: copy/pasted lines 8.3% → 12.3% (GitClear 2025, n=211M lines)



GitHub · March 2026 · 99.52% uptime

More code. More surface area for failure.

- **1 billion commits** on GitHub in 2025 — 25% YoY growth
- **26.9%** of production code is AI-authored (early 2026; n=4.2M devs)
- Honest range: **27–42%** depending on "generated" vs "assisted"
- Quality is following volume: copy/pasted lines 8.3% → 12.3% (GitClear 2025, n=211M lines)



GitHub · March 2026 · 99.52% uptime

"In six months AI would be writing 90 percent of code"

Dario Amodei · Anthropic CEO

The definition of "developer" expanded.

63% of vibe coders are non-developers

Vercel, 2026

AI would replace developers?

The definition of "developer" expanded.

63% of vibe coders are non-developers

→ PMs, designers, marketers shipping working software via v0

→ Traditional developers now ship code in domains they've **never worked in**: MCP, auth, infra glue

→ "Vibe coding has redefined who can code — most vibe coders aren't developers at all." — Vercel

~~AI would replace developers?~~

The definition of 'developer' expanded to anyone who can prompt.

When non-specialists ship code, attack surface grows.

- **30 CVEs** filed against MCP servers in 60 days (Jan–Feb 2026)
- **~⅓** of 2,614 surveyed MCP implementations susceptible to command injection
- Named production Remote Code Execution bugs in **LangFlow, LiteLLM, GPT Researcher, Flowise, Agent Zero**

When non-specialists ship code, attack surface grows

- **30 CVEs** filed against MCP servers in 60 days (Jan–Feb 2026)
- **~⅓** of 2,614 surveyed MCP implementations susceptible to command injection
- Named production Remote Code Execution bugs in **LangFlow, LiteLLM, GPT Researcher, Flowise, Agent Zero**

The gap between "easy to ship" and "safe to ship" is enormous.

The hunch is gone.

YOU WROTE IT

🚨 2:17 am — null ref in auth

auth middleware
touched Tuesday
"probably here"

token service
wrote in March

user model
Noa owns this

db layer
refactored Oct · PR#482

✓ fixed in 18 min

you wrote or reviewed every line

The hunch is gone.

YOU WROTE IT

🚨 2:17 am — null ref in auth

auth middleware
touched Tuesday
"probably here"

token service
wrote in March

user model
Noa owns this

db layer
refactored Oct · PR#482

✓ fixed in 18 min

you wrote or reviewed every line

AGENT WROTE IT

🚨 2:17 am — null ref in auth

auth middleware ?
generated Apr 3
"no idea"

token service
never read

user model
never read

db layer
never read

↻ starts from zero

you've never read any of this

FORCE 3 — LOST INTIMACY

The hunch is gone.

YOU WROTE IT

2:17 am — null ref in auth

auth middleware

"probably here"

token service

wrote in March

user model

Noa owns this

db layer

refactored Oct · PR#482

✓ fixed in 18 min

you wrote or reviewed every line

AGENT WROTE IT

2:17 am — null ref in auth

auth middleware

no idea

token service

never read

user model

never read

db layer

never read

⟳ starts from zero

you've never read any of this

> *"When code breaks, you are not reconstructing intent. You are reverse-engineering a decision process that was never explained and no longer exists."*
>
> — David Monnerat

← → navigate · N = speaker notes · ESC close

The codebase is accumulating decisions that nobody made.

You can fight bad outputs. You can't fight the volume curve.

- Ranting about "AI slop" doesn't reduce slop — the volume ships either way
- The choice: whether you have the **operational muscle** to run code you didn't write

You can fight bad outputs. You can't fight the volume curve.

- Ranting about "AI slop" doesn't reduce slop — the volume ships either way
- The choice: whether you have the **operational muscle** to run code you didn't write

Where do we go from here?

Managing without reading.

- You've managed code you didn't write before — every tech lead has
- Trust, but verify → **Trust, but make it prove it**
- Your job: defining what good behaviour looks like, detecting deviation

YOUR JOB

1. Define what good behaviour looks like

2. Detect deviation

WHAT AGENTS NEED

1. Goal

2. A way to verify their work

You've done this before. The verbs just change object.

YOU DID THIS (HUMANS)

YOU'LL DO THIS (AGENTS)

You've done this before. The verbs just change object.

YOU DID THIS (HUMANS)	YOU'LL DO THIS (AGENTS)
Hire engineer	Spin up agent

You've done this before. The verbs just change object.

YOU DID THIS (HUMANS)	YOU'LL DO THIS (AGENTS)
Hire engineer	Spin up agent
Set expectations & 1:1s	Define Goals, SLOs & instrument

You've done this before. The verbs just change object.

YOU DID THIS (HUMANS)	YOU'LL DO THIS (AGENTS)
Hire engineer	Spin up agent
Set expectations & 1:1s	Define Goals, SLOs & instrument
Code reviews	Behavioural reviews via traces / metrics

You've done this before. The verbs just change object.

YOU DID THIS (HUMANS)	YOU'LL DO THIS (AGENTS)
Hire engineer	Spin up agent
Set expectations & 1:1s	Define Goals, SLOs & instrument
Code reviews	Behavioural reviews via traces / metrics
Performance evaluations	SLO compliance reports

You've done this before. The verbs just change object.

YOU DID THIS (HUMANS)	YOU'LL DO THIS (AGENTS)
Hire engineer	Spin up agent
Set expectations & 1:1s	Define Goals, SLOs & instrument
Code reviews	Behavioural reviews via traces / metrics
Performance evaluations	SLO compliance reports

MANAGE BY OUTCOME

You've done this before. The verbs just change object.

YOU DID THIS (HUMANS) | YOU'LL DO THIS (AGENTS)

Hire engineer | Spin up agent

Set expectations & KPIs | Define Goals, SLOs & instrument

Code reviews | Behavioural reviews via traces / metrics

Performance evaluations | SLO compliance reports

MANAGE BY OUTCOME

Stop reviewing *code.*

Start reviewing observable *behavior.*

Scenario: agent ships a payment feature.

WHAT THE AGENT SHIPS

- ✅ tests pass (47/47)
- ✅ no lint errors
- ✅ coverage 89%
- ✅ ready to merge

Scenario: agent ships a payment feature.

WHAT THE AGENT SHIPS

✅ tests pass (47/47)

✅ no lint errors

✅ coverage 89%

✅ ready to merge

WHAT YOU DEMAND INSTEAD

→ Add metrics: payment.retry.attempts, payment.retry.exhausted

→ Run synthetic failure in staging

→ Show me the trace + metric

Scenario: auth endpoint.

```
p99 latency < 300ms under expected load
error rate < 0.1%, no single type > 50% of failures
token refresh success rate > 99.5%
≤ 5 failed logins per IP/min before rate-limit kicks
in
100% of auth events emit structured log: user-id +
```

Four shifts.

Four shifts.

1. Observability-first deployment ← *the test*
2. SLOs as the production contract
3. Policy-as-code guardrails ← *the assertions*
4. AI-generated investigation reports

Observability-first.

- Instrument **before you ship** — it's the deploy gate, the unit-test of production
- "Agent-observable code" = structured logs, semantic metric labels, trace span attributes that explain **intent**

Observability-first.

- Instrument **before you ship** — it's the deploy gate, the unit-test of production
- "Agent-observable code" = structured logs, semantic metric labels, trace span attributes that explain **intent**

NOT INTENT

```
ERROR: failed
```

INTENT

```
payment.retry.exhausted
user=42 reason=upstream_timeout
attempt=4 budget=3
```

Observability-first.

- Instrument **before you ship** — it's the deploy gate, the unit-test of production
- "Agent-observable" code: structured logs, semantic event names, span attributes that explain **intent**

> *Code is agent-observable if a person (or an agent) can reason about its behaviour without reading the source. That means structured logs with semantic event names, metrics with documented units and labels, traces with attributes that explain intent.*

NOT INTENT

ERROR: failed

INTENT

payment.retry.exhausted

rc=42 reason=upstream_timeout

attempt=4 budget=3

SLOs as the contract.

- The SLO defines what **"working"** means — it's how the agent verifies its own work
- **Pre-deploy:** agent must prove the SLO holds under load — not "tests pass"
- **Post-deploy:** SLO compliance overrides every other signal, including the test suite

SLOs as the contract.

- The SLO defines what "**working**" means — it's how the agent verifies its own work
- **Pre-deploy:** agent must prove the SLO holds under load — not "tests pass"
- **Post-deploy:** SLO compliance overrides every other signal, including the test suite

> SLOs turn your intent from the vibe of just *"Working"* to being a **measurable contract**.

"PRODUCTION IS WHERE THE RIGOR GOES." — CHARITY MAJORS

> As a manager you set OKRs for your employees.
>
> As an **agent manager** you set SLOs.

Policy-as-code guardrails.

- The deploy gate is **code** — version-controlled, enforced by CI
- **Common gates:** latency budget, error rate, dependency health, security scan, IaC compliance
- Agent-written code passes through the **same gate** as human code — automatically

WHAT'S NEXT

Emerging Policy-as-Prompt research and tools

Tooling exists — ecosystem is early

Start from the investigation report.

OLD: TAIL LOGS

```
$ tail -f /var/log/app.log | grep
ERROR
[03:11:58] ERROR req_id=a4f2
status=500
[03:11:59] ERROR req_id=a4f3
status=500
[03:12:00] ERROR req_id=a4f4
status=500
[03:12:01] ERROR req_id=a4f5
status=500
```

Start from the investigation report.

OLD: TAIL LOGS

```
$ tail -f /var/log/app.log | grep ERROR
[03:11:58] ERROR req_id=a4f2 status=500
[03:11:59] ERROR req_id=a4f3 status=500
[03:12:00] ERROR req_id=a4f4 status=500
[03:12:01] ERROR req_id=a4f5 status=500
```

NEW: AI INVESTIGATION REPORT

```
// AI Investigation Report — 03:12 UTC
Latency spike at 02:47:03 UTC →
deploy cart-service@7f3a91
N+1 query in
OrderHistoryService.getRecentOrders()
Fix: prefetch_related('items') line 47
Confidence: HIGH
[git blame]
```

— Real today: PagerDuty SRE Agent (Oct 2025), incident.io (Jul 2025), Azure AI SRE (May 2025 preview)

Start from the investigation report.

OLD: TAIL LOGS

```
$ tail -f /var/log/app.log | grep ERROR
[03:11:58] ERROR req_id=a4f2
status=500
[03:11:59] ERROR req_id=a4f3
status=500
[03:12:00] ERROR req_id=a4f4
status=500
[03:12:01] ERROR req_id=a4f5
status=500
```

NEW: AI INVESTIGATION REPORT

```
// AI Investigation Report — 03:12 UTC
Latency spike at 02:47:03 UTC →
deploy cart-service@7f3a91
N+1 query in
OrderHistoryService.getRecentOrders()
Fix: prefetch_related('items') line
47
Confidence: HIGH
[git blame]
```

Use your AI bullshit meter →

— Real today: PagerDuty SRE Agent (Oct 2025), incident.io (Jul 2025), Azure AI SRE (May 2025 preview)

If the report is wrong, ask three questions:

1. Is *my* instrumentation right?
2. Is *my* analysis right?
3. Did the agent have enough context?

The new on-call skill.

OLD

Read the code.

Find the bug.

Know every service.

The new on-call skill.

OLD

✕ ~~Read the code.~~ — Force 1: volume is exponential

Find the bug.

Know every service.

Operational metrics matter more, not less: MTTR, MTBF, SLO compliance, error budget burn rate.

The new on-call skill.

OLD

- ~~Read the code.~~ — Force 1: volume is exponential
- ~~Find the bug.~~ — Force 2: breadth spans stacks you don't own
- Know every service.

Operational metrics matter more, not less: MTTR, MTBF, SLO compliance, error budget burn rate.

The new on-call skill.

OLD

- ✕ ~~Read the code.~~ — Force 1: volume is exponential
- ✕ ~~Find the bug.~~ — Force 2: breadth spans stacks you don't own
- ✕ ~~Know every service.~~ — Force 3: lost intimacy, code you never wrote

Operational metrics matter more, not less: MTTR, MTBF, SLO compliance, error budget burn rate.

THINGS TO BE GOOD AT

1. Reading distributed traces
2. Recognising failure-mode patterns across stacks you didn't write
3. Validating AI hypotheses
4. Knowing when to escalate vs. when to trust the report

The reframe.

"You build it, you run it."

⇕

"You ship agents, you own their outcome."

Three questions before every agent-written deploy.

01

What does good behaviour look like?

Have you defined the SLOs?

02

How will I know if it's misbehaving?

Is the instrumentation in place?

03

Who bears the responsibility?

Is responsibility clear?

AWS X DEVOPS · 2026

Your job isn't writing/reading code. It's writing/reading systems.

Ran Tavory · @rantav

AWS X DEVOPS · 2026

Your job isn't writing/reading code. It's writing/reading systems.

Ran Tavory · @rantav

Thank you.