7 Ways to Fail at Building a Platform

Coté – DevOps Meetup Amsterdam, November 10th, 2026.


HA HA!
BUSINESS!


Rewriting
Everything
in Rust




Rewriting
Everything
in Rust





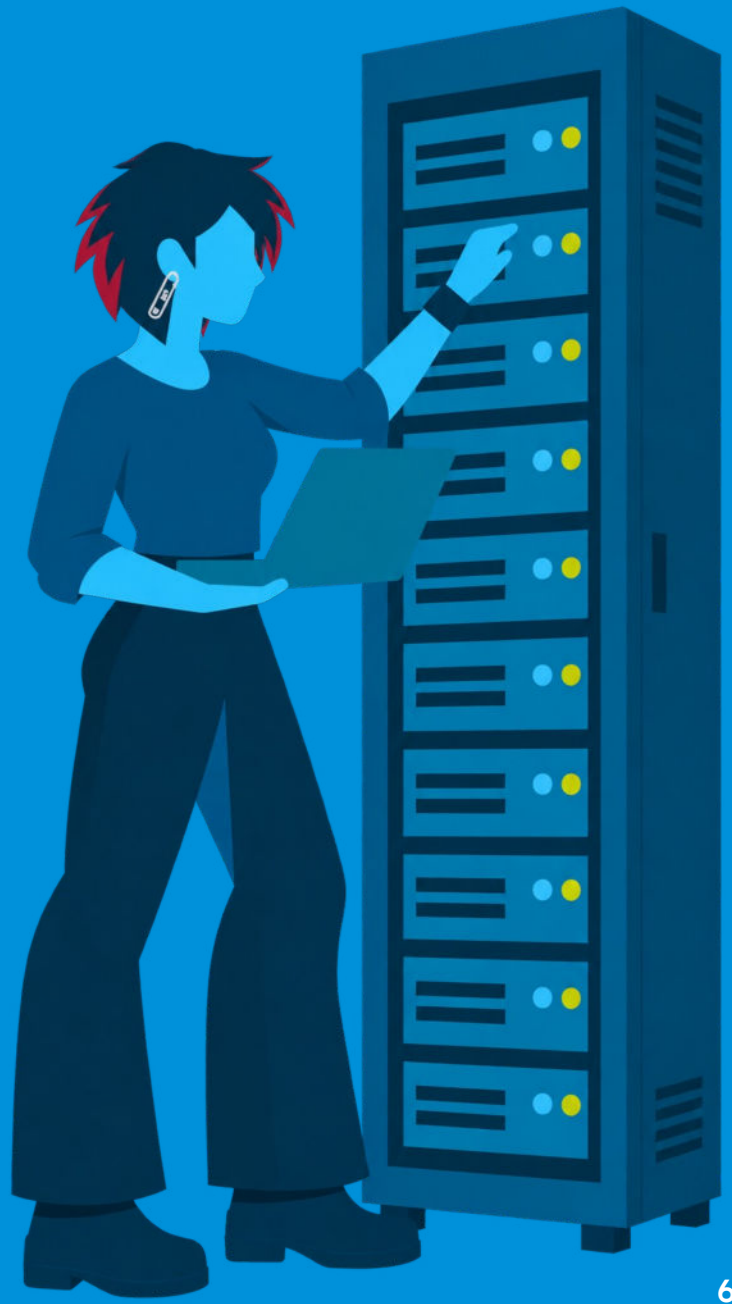


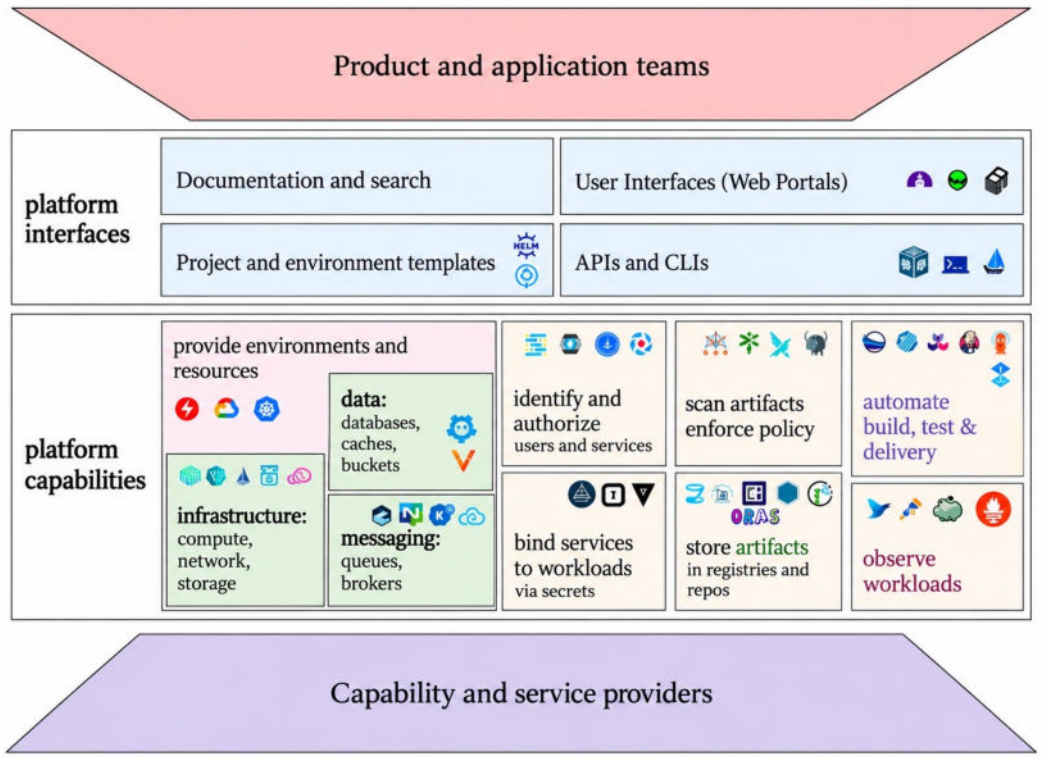


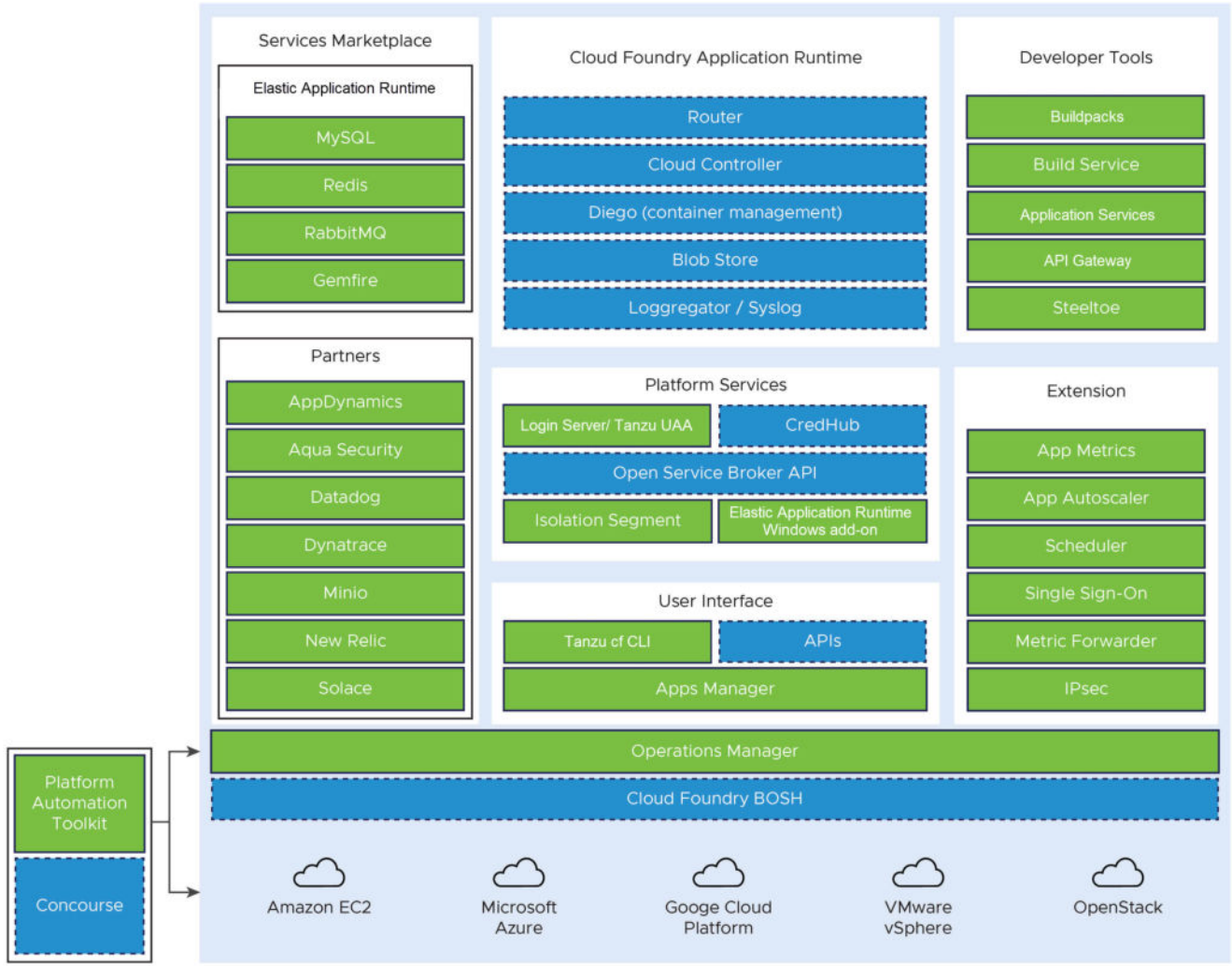


> [B]y 2027, 80% of large organizations will embrace platform engineering to successfully scale DevOps initiatives in hybrid cloud environments – up from less than 30% in 2023." Gartner, 2025.

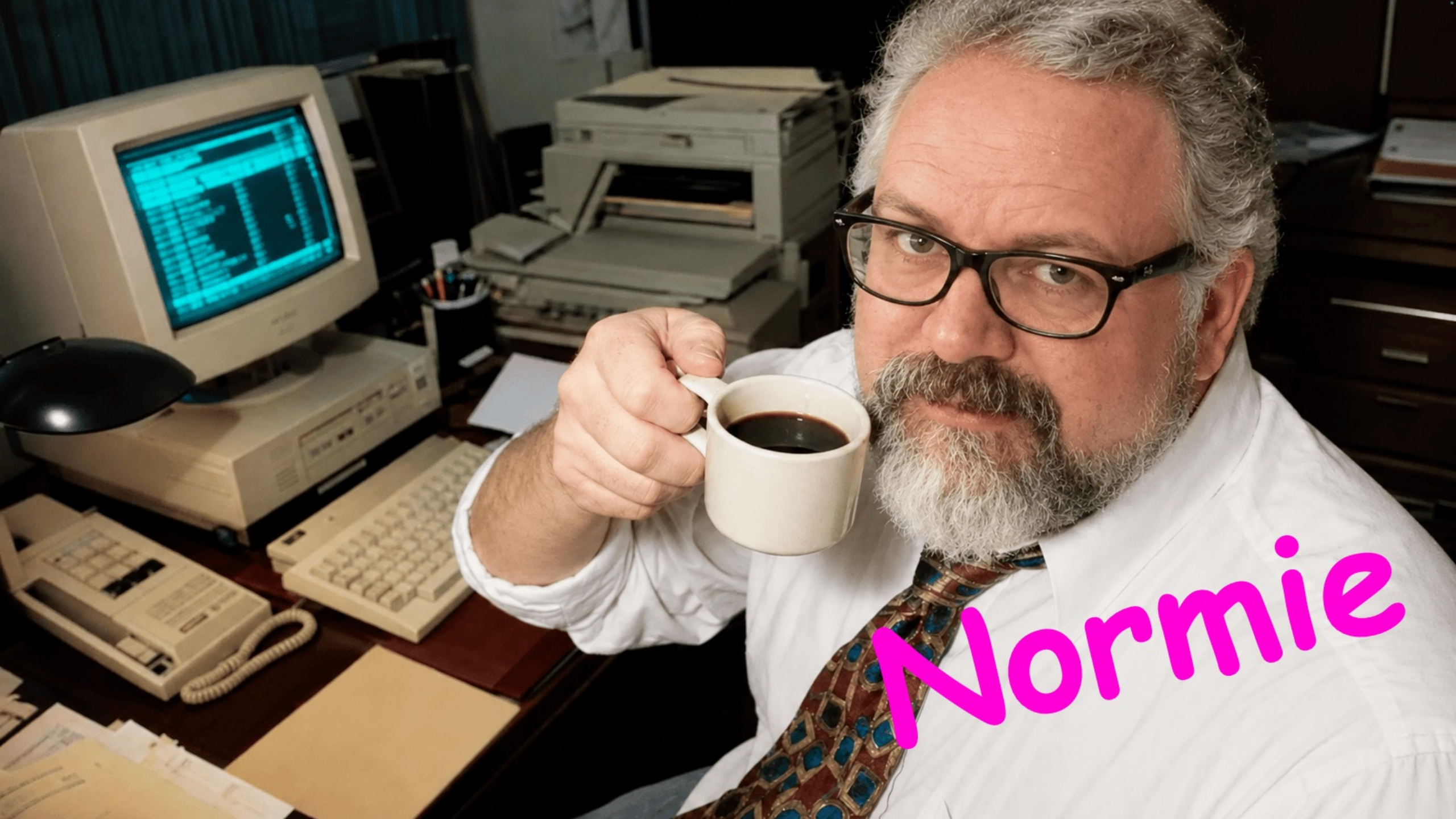
> 76% of organizations have at least one dedicated platform" DORA, 2025.



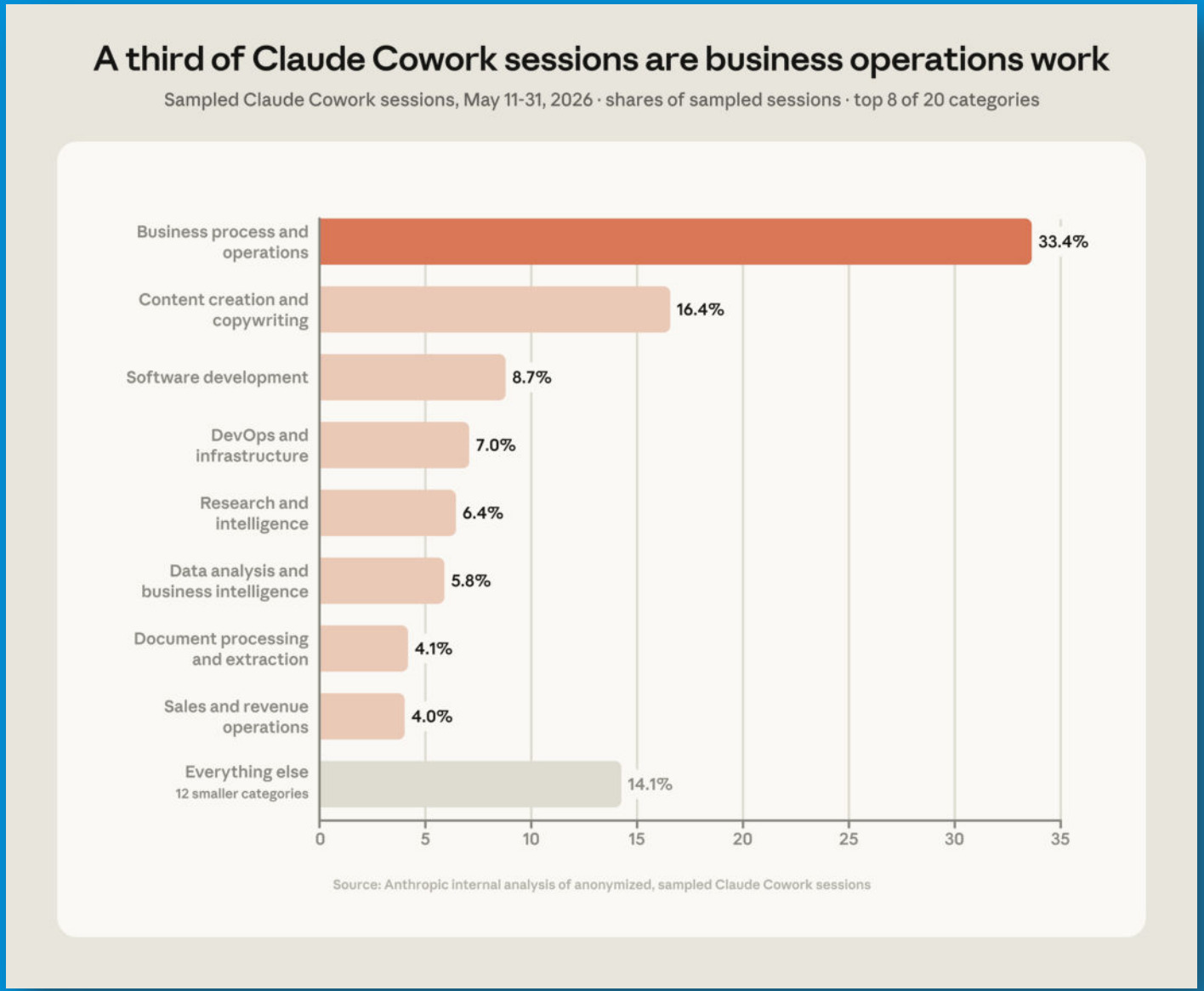




Sources: "CNCF Platforms White Paper," March 2023; Tanzu Platform 11 documentation.


Normie

Yes…



Source: "How people are using Claude Cowork," Anthropic, July 7th, 2026, from "1.2 million anonymized and aggregated Claude Cowork sessions from May 11–31, 2026."

But…

"Accenture is trying to figure out how to stop non-technical workers from blowing through companies' AI token budget on trivial tasks like **converting PDFs to presentation slides**."

404 Media, June 2026

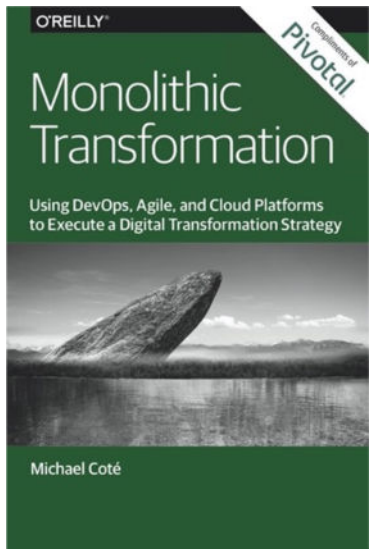


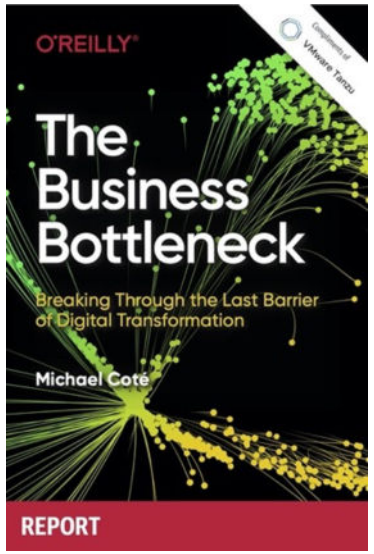


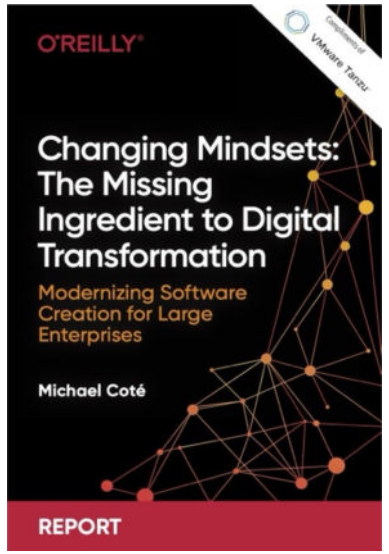
Coté

https://www.cote.io/

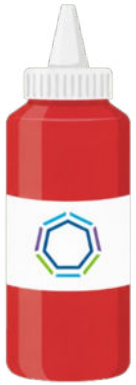
michael.cote@broadcom.com











Tanzu Catsup

RedMonk DELL 451 Research S&P Global Market Intelligence Pivotal vmware Tanzu by Broadcom

WHITE PAPER
August 2025

The Upside-Down Economics of DIY PaaS

Seven pitfalls of building your own platform

vmware by Broadcom

WHITE PAPER: **June 2026**



Why You Should Treat Your Platform as a Product

vmware by Broadcom

#1
Developer-Built Platforms.

Or, “The Day Two Hangover”

Source: “From Platform Sprawl to Producer-Consumer: Lessons from Veeam’s Day Two Reality,” Syntasso, April, 2026.





Let's build a platform!



Generated with ChatGPT on June 23rd, 2026 and July 8th, 2026

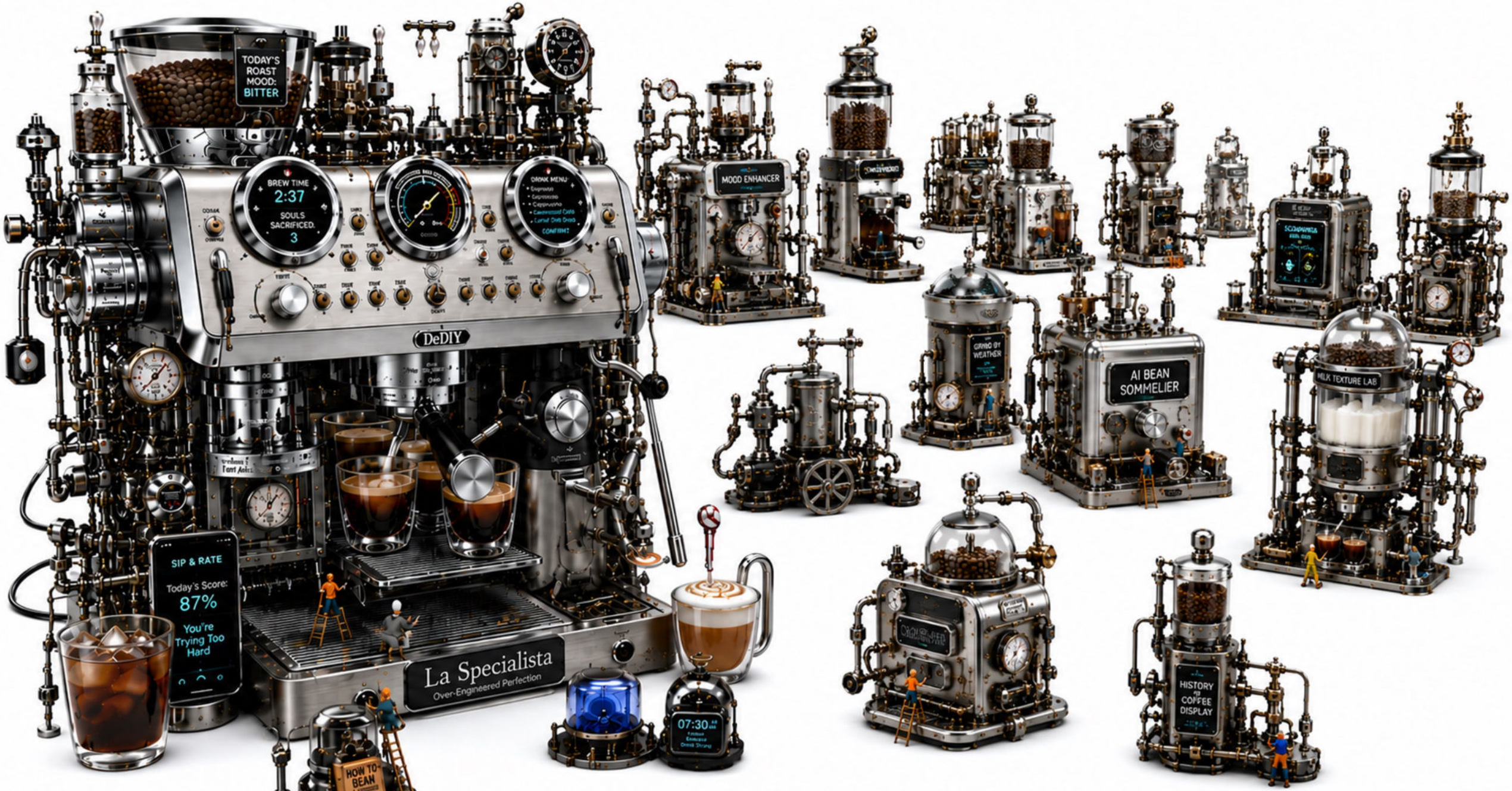




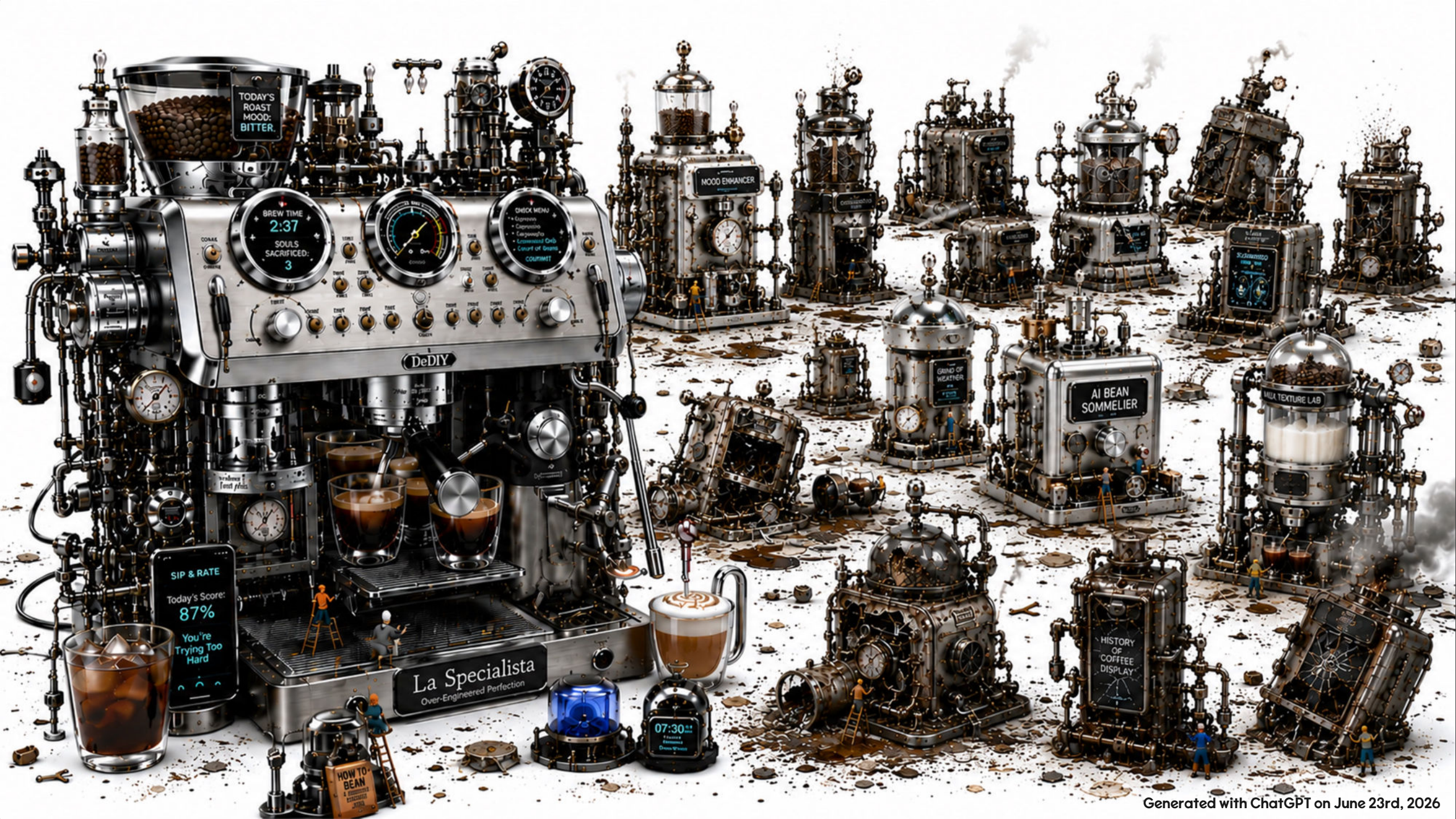
Let's build a platform!



Generated with ChatGPT on June 23rd, 2026 and July 8th, 2026



Generated with ChatGPT on June 23rd, 2026

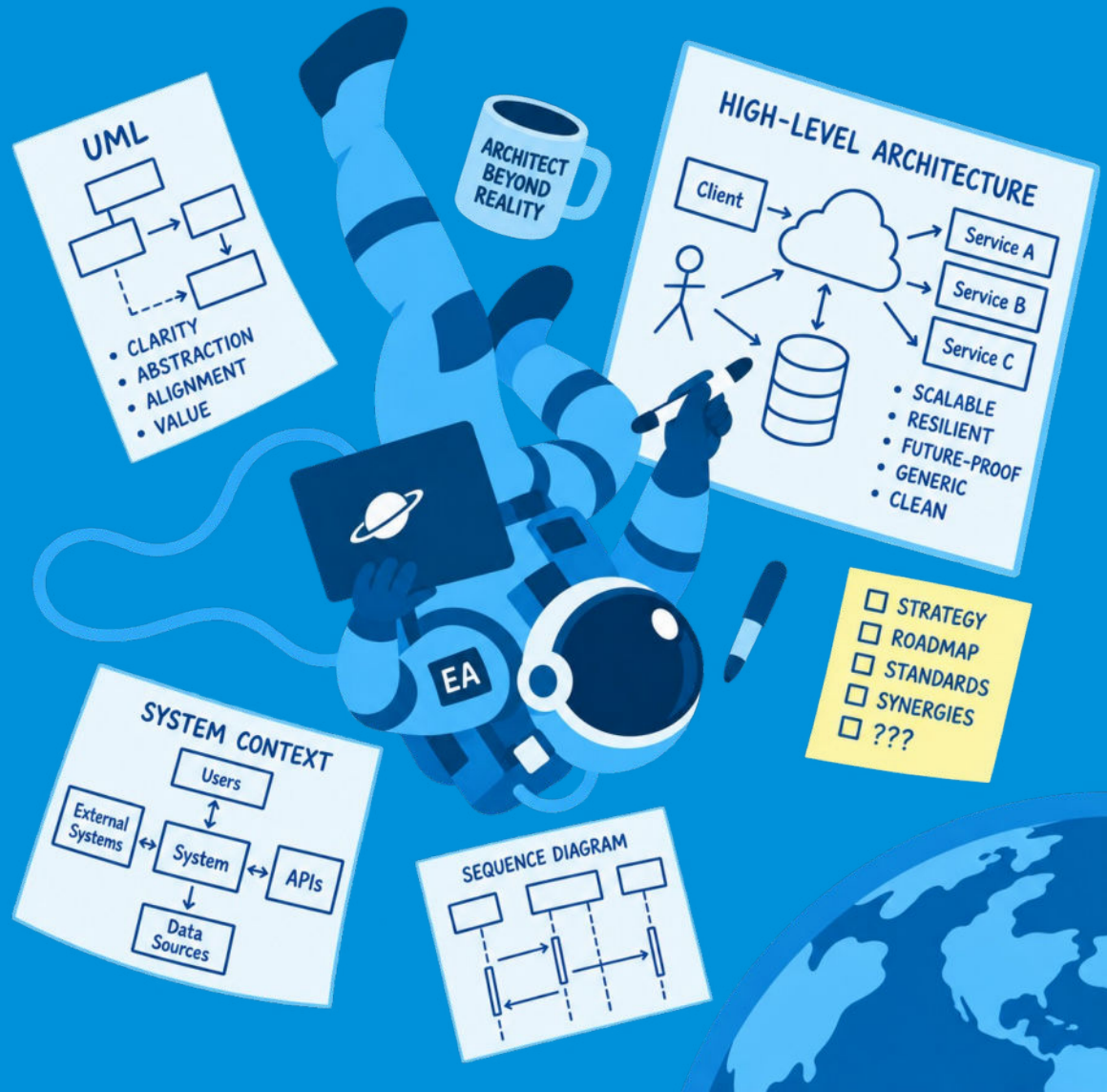


Generated with ChatGPT on June 23rd, 2026

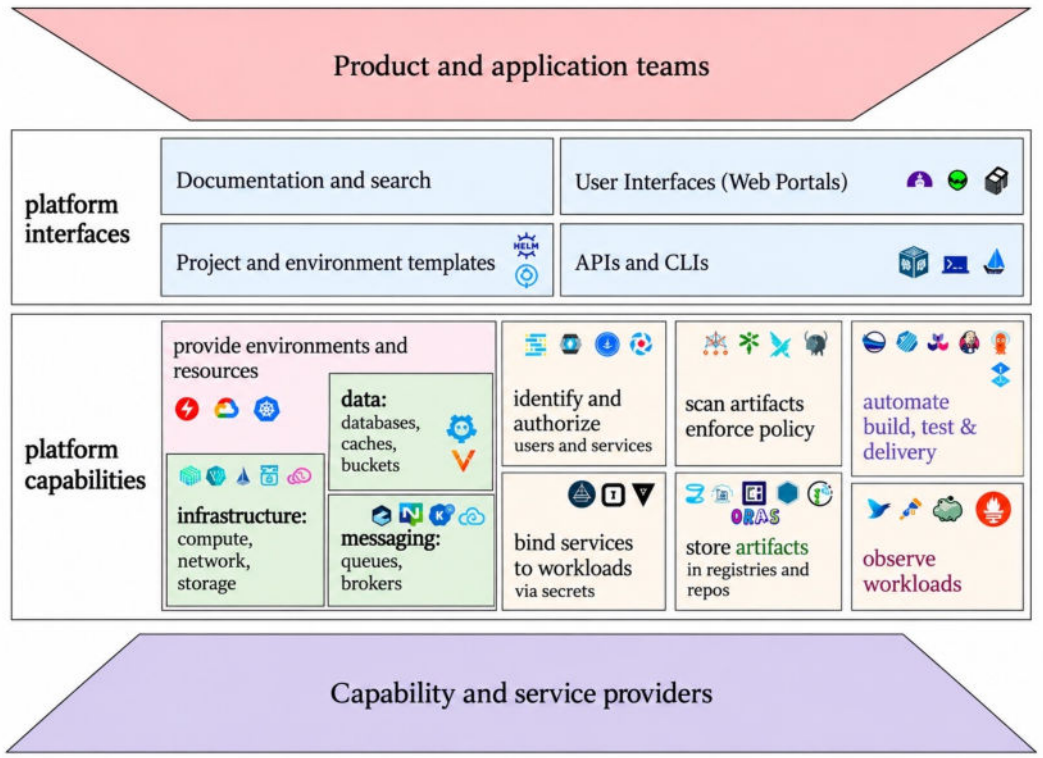


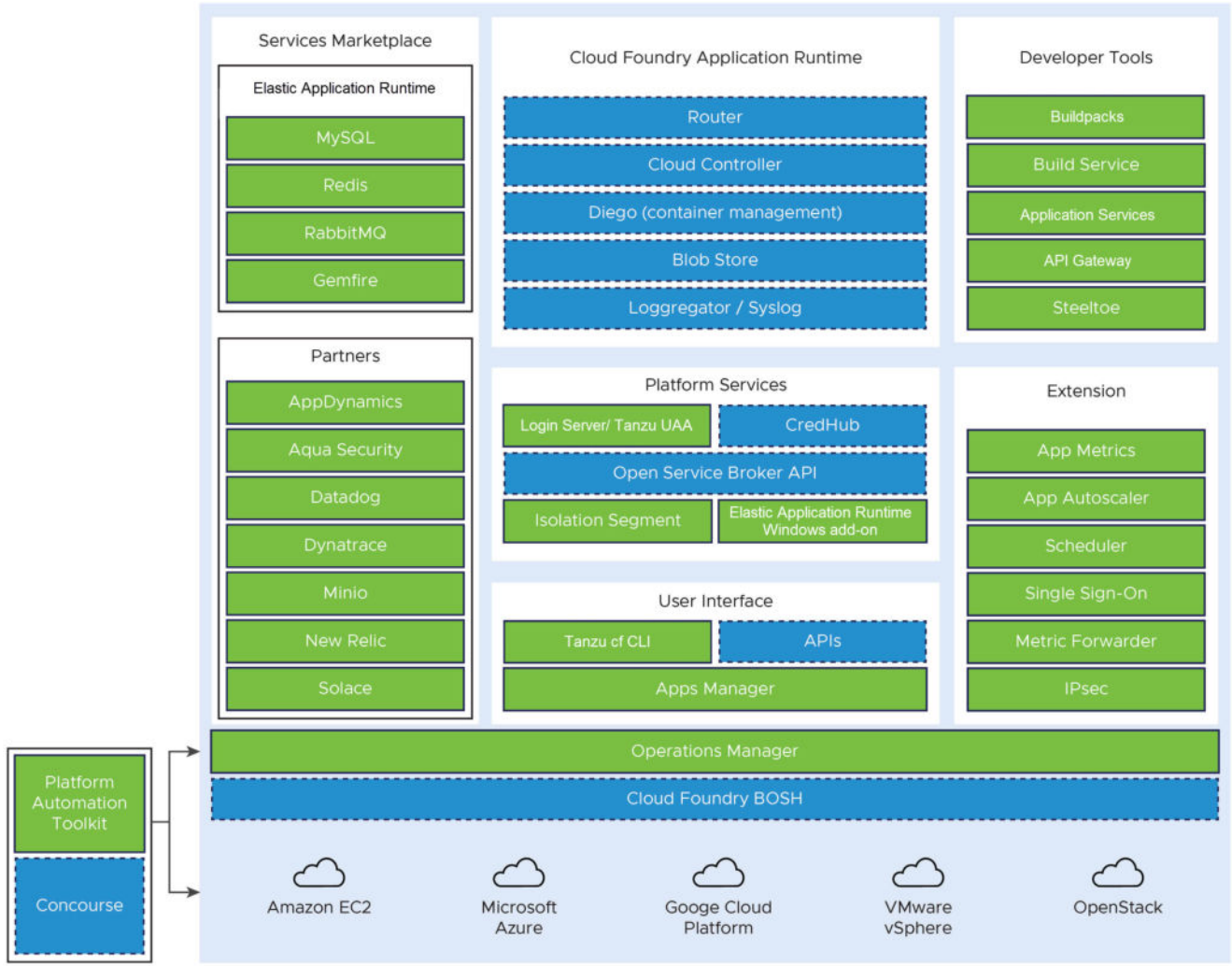
Not my mistake, now my problem.

CoE it from orbit



#2
Unexpected scope creep.





Sources: "CNCF Platforms White Paper," March 2023; Tanzu Platform 11 documentation.

More than namespaces, yaml templates, & base container images

- App delivery.
- Backup and restore.
- Patch management.
- Observability, logs, monitoring.
- Service management & use.
- RBAC, etc.
- Vulnerability scanning.
- Dev framework integration.
- High availability & the other –ility's.
- Multi-region deployment.
- Sovereign cloud.
- Auditing and compliance.
- Multi-tenancy.
- Upgrading the platform.
- Gateways, brokers, load balancers.
- CI/CD, itself or integration.

See also "Stop Renting Your Knowledge," Cur8s, Adib Saikali, Winter, 2025.

	Level 1 – Build	Level 2 – Operate	Level 3 – Scale	Level 4 – Improve	Level 5 – Adapt
Primary focus	First cloud native workloads	Production + standardization	Org-wide repeatability	Governance + security by default	Continuous optimization
Platform & infrastructure	Initial Kubernetes clusters Basic container registry Infrastructure as Code with Terraform	GitOps for apps via Argo CD or Flux Thin internal developer platform (templates, self-service) Centralized environments	Multiple standardized clusters (multi-region / multi-env) Formal Internal Developer Platform (portals, templates, APIs) GitOps extended to platform components	Drift detection + auto-remediation Zero-trust platform access Security posture management integrated with ops	Policy-driven workload placement (cloud/on-prem/edge) Platform continuously reshaped based on usage data
Application delivery	Helm or raw manifests via Helm Basic CI pipelines 1–2 pilot apps	Helm + Kustomize at scale Runtime config via ConfigMaps/Secrets Standard base images + scanning + SBOMs	Artifact signing + automated promotion pipelines Namespaces-as-a-Service Horizontal + event-driven autoscaling from app metrics	End-to-end supply chain security (signed images, enforced SBOMs) Admission controls everywhere	Progressive delivery (canaries, feature flags everywhere)Predictive scaling from production signals
Observability & operations	Minimal logging/metrics (often cloud defaults + early Prometheus)	Metrics + logs + early tracing using OpenTelemetry Central log aggregation	Distributed tracing as first-class signal Automated backup, DR, cluster lifecycle	Central audit pipelines (SCM, CI, clusters, apps) Runtime policy enforcement	Automated performance/reliability feedback loops Anomaly detection from live telemetry
Networking & security	Manual secrets Basic perimeter security	Admission controls Container/runtime scanning Early service mesh (often built on Envoy)	Operational service mesh (mTLS, retries, traffic shaping) Multi-tenancy with quotas	Workload identity + automated cert rotation Fine-grained authZ via mesh Zero-trust networking	AI-assisted remediation Continuous security optimization
Cost & optimization	Mostly manual cost awareness	Initial resource limits Early FinOps signals (namespace quotas, requests/limits)	Chargeback/showback per team or namespace	Integrated FinOps dashboards + budget controls	FinOps directly drives autoscaling and scheduling Continuous cost optimization
AI (where applicable)	Mostly experimental	First production models Basic observability + access controls			

Sources: "Platform Engineering Maturity Model," CNCF Platforms Working Group, October, 2023. See also "Cloud Native App Platforms: New Research Shows Struggles and Hope," Camille Crowell-Lee and Rita Manachi, June, 2024. See also beta version of Cloud Native Maturity Model: Core content updates for v4.0 (#84) (6e03f6c).

“Most rearchitectures take three to five years” – all apps even longer

Scrappy Platform	Run Scr	ild new platform, migrate apps
1 year	2 to 3 yea	3 to 5 years

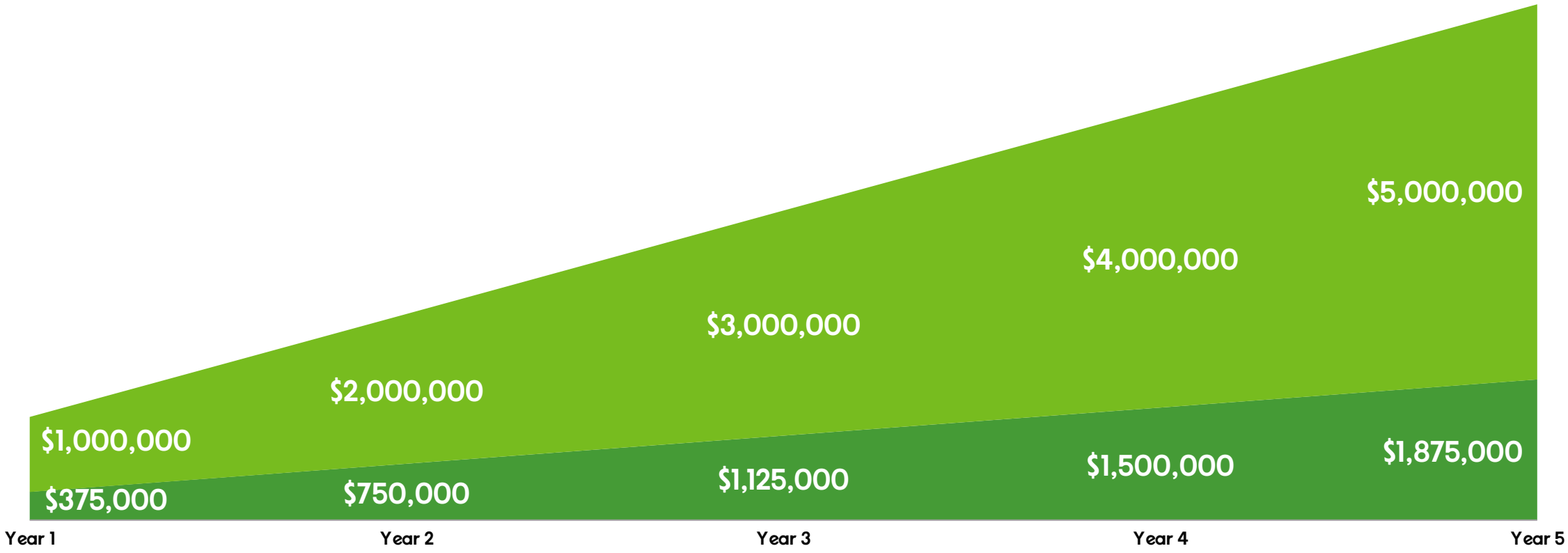
Assuming it works!!



Sources: customer interviews; ”over drinks” confessionals; *Platform Engineering*, Fournier & Nowland, 2024; “The Upside-Down Economics of DIY PaaS,” Coté, Jared Ruckle, Bryan Friedman, Matt Walburn, 2026.

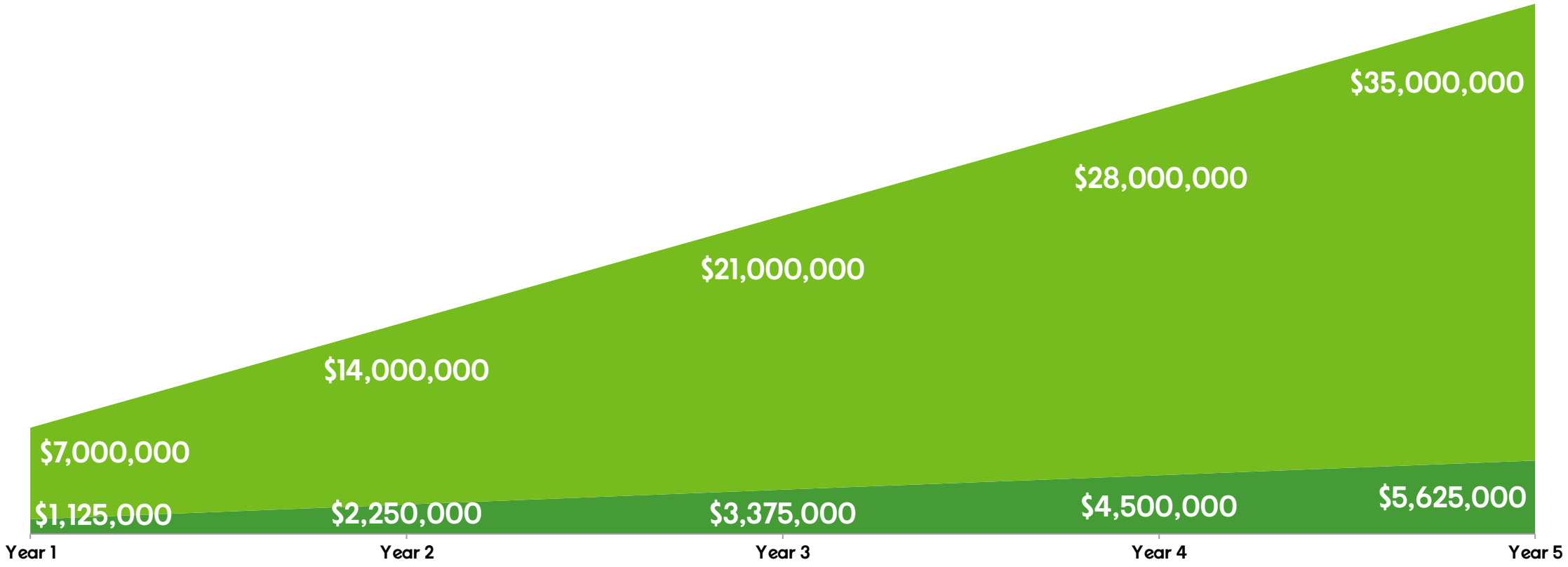
#4
Underestimating the ongoing investment

Cumulative
Platform Salary Spend
One team of 3 to 8 people, annually



Sources: "The Upside-Down Economics of DIY PaaS," August, 2025; "How much does it cost to build an internal developer platform?" Tanzu Catsup, January 27th, 2026. Conversations with FSIs.

Cumulative Platform Salary Spend 3 to 8 teams, annually



Sources: "The Upside-Down Economics of DIY PaaS," August, 2025; "How much does it cost to build an internal developer platform?" Tanzu Catsup, January 27th, 2026. Conversations with FSIs.

- 30,000 devs / 50 ops
- 6,500 devs/16 ops
- 2,500 devs / 5 ops
- 1,200 devs / 6 ops

- 350 apps / 7 ops
- 300 apps / 8 ops

- 45 app teams / 5 ops
- 300 app teams/ 4 ops

Sources: Kroger, GAIC, Mercedes-Benz, conversations with FSI platform engineers; "Enterprise Grade Platform Engineering at Charles Schwab," Coté, September, 2024, based on Schwab's Explore 2024 panel; Rabobank ops conversations, CF Day EU 2025, Oct 7th, 2025; "3 Cloud Foundry Stories," Coté, CF Day EU, Oct 7th, 2025.

#5
Platform as a project

“We are building this platform not for us, we are building it for Mercedes-Benz developers.”

Thomas Müller, Mercedes-Benz



More on product management: “Why You Should Treat Your Platform as a Product,” June, 2026; “The day 2 hangover,” Cat Morris & Nitish Malhotra, PlatformCon, June 26th, 2026.

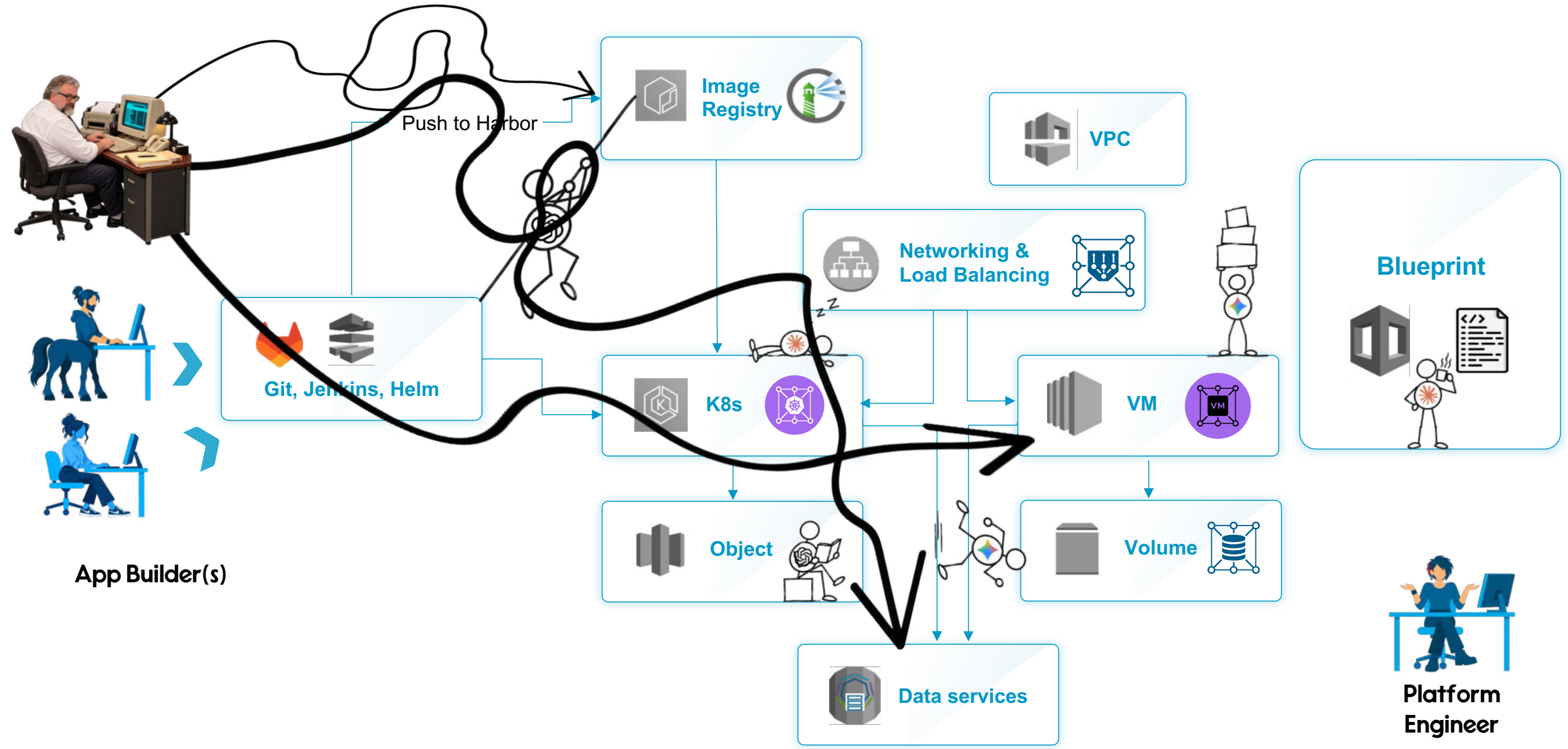
Find the Developer Toil, Confusion, Blockers

Find the Developer Toil, Confusion, Blockers

- What are we making?
- We have a strong vision for our product, and we're doing important work together every day to fulfill that vision.
- I have the context I need to confidently make changes while I'm working.
- I am proud of the work I have delivered so far for our product.
- I am learning things that I look forward to applying to future products.
- My workstation seems to disappear out from under me while I'm working.
- It's easy to get my workstation into the state I need to develop our product.
- What aspect of our workstation setup is painful?
- It's easy to run our software on my workstation while I'm developing it.
- I can boot our software up into the state I need with minimal effort.
- What aspect of running our software locally is painful? What could we do to make it less painful?
- It's easy to run our test suites and to author new ones.
- Tests are a stable, reliable, seamless part of my workflow.
- Test failures give me the feedback I need on the code I am writing.
- What aspect of production support is painful?
- We collaborate well with the teams whose software we integrate with.
- When necessary, it is within my power to request timely changes from other teams.
- I have the resources I need to test and code confidently against other teams' integration points.
- What aspect of integrating with other teams is painful?
- I'm rarely impacted by breaking changes from other tracks of work.
- We almost always catch broken tests and code before they're merged in.
- What aspect of committing changes is painful?
- Our release process (CI/CD) from source control to our story acceptance environment is fully automated.
- If the release process (CI/CD) fails, I'm confident something is truly wrong, and I know I'll be able to track down the problem.
- What aspect of our release process (CI/CD) is painful?
- Our team releases new versions of our software as often as the business needs us to.
- We are meeting our service-level agreements with a minimum of unplanned work.
- When something is wrong in production, we reproduce and solve the problem in a lower environment.

Sources: "Developer Toil: The Hidden Tech Debt," Susie Forbath, Tyson McNulty, and Coté, August, 2022. See also Michael Galloway's interview questions for platform product managers.

Agents in your CaaS



Source: "Inside the Coding Agent-Powered SDLC," Adib Saikali & Frank Denneman, Explore 2026.

Guardrails at build time with a proper platform









App Builder(s)

cf push
cf bind-service
cf start
etc…



buildpack



Platform as a Service

build

app

app serving

service create

service bind

registry





Platform Engineer

VMware Tanzu Platform

Sources: "Inside the Coding Agent-Powered SDLC," Adib Saikali & Frank Denneman, Explore 2026; "When everyone can build: The platform behind 1000 citizen-built app," Boyan Dimitrov, CTO of SIXT, PlatformCon 2026, published 22 June 2026.

Agent *Provisions* Infrastructure

- Agent must reason about infrastructure topology, sizing, and configuration
- Each agent can make different provisioning decisions
- Dedicated infrastructure can lead to over-provisioning and resource sprawl
- More agent decisions create more opportunities for errors



Agent *Consumes* Services

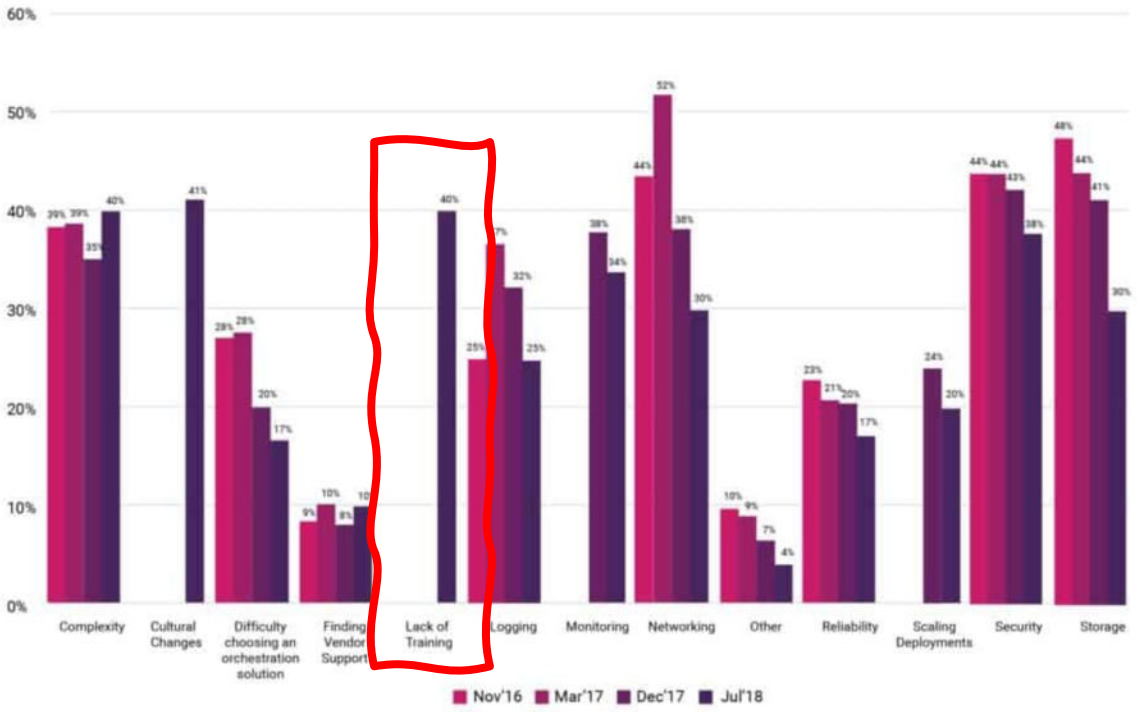


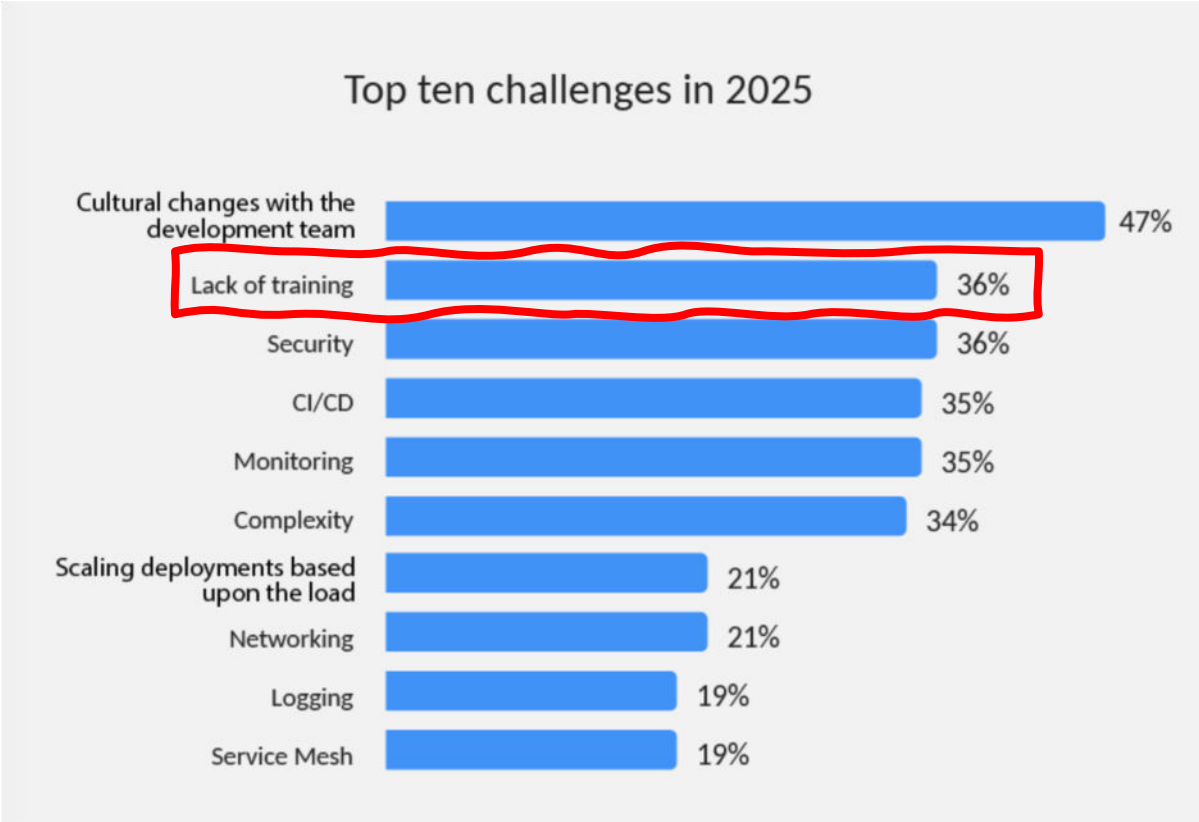
- Agent selects from platform-defined service plans
- Platform team controls sizing, configuration, policy, and lifecycle
- Services can be backed by shared, right-sized infrastructure
- Every agent gets a repeatable, governed service

Source: "Inside the Coding Agent-Powered SDLC," Adib Saikali & Frank Denneman, Explore 2026.

#6
Retaining skilled people

Lack of training, 2017 to 2025





40% in 2017

36% in 2025

Sources: CNCF Survey, August, 2018; CNCF Survey, January, 2026 (for 2025).

#7
Resume-driven development

Résumé-Driven Development: A Definition and Empirical Characterization

Jonas Fritzsch, Marvin Wyrich, Justus Bogner, Stefan Wagner
University of Stuttgart, Germany, Institute of Software Engineering
{firstname.lastname}@iste.uni-stuttgart.de

.SE] 29 Jan 2021

***Abstract*—Technologies play an important role in the hiring process for software professionals. Within this process, several studies revealed misconceptions and bad practices which lead to suboptimal recruitment experiences. In the same context, grey literature anecdotally coined the term *Résumé-Driven Development* (RDD), a phenomenon describing the overemphasis of trending technologies in both job offerings and resumes as an interaction between employers and applicants. While RDD has been sporadically mentioned in books and online discussions, there are so far no scientific studies on the topic, despite its potential negative consequences. We therefore empirically investigated this phenomenon by surveying 591 software professionals in both hiring (130) and technical (558) roles and identified RDD facets in substantial parts of our sample: 60% of our hiring professionals agreed that trends influence their job offerings, while 82% of our software professionals believed that using trending technologies in their daily work makes them more attractive for prospective employers. Grounded in the survey results, we conceptualize**

dents [...] learn a new technology at least every few months or once a year" [4]. The survey also revealed that technologies are regarded as the most important job factor by software professionals. This focus may come at the expense of other skills, leading to *"insufficient development competences"* [2] as Ebert and Counsell state it.

In the context of the software professional recruiting process, lively discussions have come up in developer forums, blogs, or social media where occasionally the terms *Résumé-Driven Development* (RDD) [5], [6], [7] and similarly *CV-Driven Development* [8] were used. Classon associates this notion with selecting *"tech stacks, architecture, methodologies, and protocols based on what looks good on the resume"* [9], while Ford et al. describe it as a pitfall by architects becoming enamored in latest technologies [10]: *"utilizing every frame-*

"public's decreasing acceptance of [...] self-driving cars" [2]. Already a decade ago, Adomavicius et al. regarded the sheer number of available technologies and the complex set of relationships among them as a major challenge [3].

The recent Stack Overflow 2020 Developer Survey among 65.000 software developers found that *"around 75% of respon-*

To refine our objective and to set the scope for a more detailed investigation, we formulate several research questions. In this context, we distinguish between a hiring and an applicant perspective. While software professionals sometimes act in both roles, hiring professionals and computer science students are commonly associated exclusively with one perspective.

phenomenon by surveying 591 software professionals in both hiring (130) and technical (558) roles and identified RDD facets in substantial parts of our sample: 60% of our hiring professionals agreed that trends influence their job offerings, while 82% of our software professionals believed that using trending technologies in their daily work makes them more attractive for prospective employers. Grounded in the survey results, we conceptualize a theory to frame and explain Résumé-Driven Development.

additional language or framework. Moreover, introducing new technologies implies a learning curve and may come with maturity issues that impact reliability. Extensive RDD-based technology selection may therefore lead to complex or even unmaintainable software consisting of technologies which are not suitable for the requirements, which are unfamiliar to current or future employees, or which did not deliver on their promise and were discontinued.

may impact maintainability: technologies need to be regularly updated, dependencies have to be managed, and knowledge sharing efforts among team members increase with every additional language or framework. Moreover, introducing new technologies implies a learning curve and may come with maturity issues that impact reliability. Extensive RDD-based technology selection may therefore lead to complex or even unmaintainable software consisting of technologies which are not suitable for the requirements, which are unfamiliar to current or future employees, or which did not deliver on their promise and were discontinued.

Second, RDD can lead to false expectations and disappointment in the recruiting process on both sides. In a recent HackerRank survey among 71,281 developers [32], the top answer for *"What turns developers off from employers?"* with an affirmation of more than two-thirds was *"not enough clarity on role or where I'll be placed"*. Similarly, Behroozi at al. [21] identified a number of suboptimal practices which may sabotage recruitment processes in a study of over 10,000 Glassdoor reviews. A prevalent theme among them were inadequately communicated criteria by HR. RDD-based hiring can lead to similar frustrations. A strong focus on technologies during hiring may also lead to the neglection of other important skills for creating high-quality software products in a team. This is reinforced by nine *hiring* professionals who provided free-text comments about applicant traits they value much more than experience with trending technology, e.g. soft skills like communication, self-motivation, the willingness to learn, being a cultural fit for the team, or an understanding of the fundamental principles behind technologies. Since high employee

minimal in the software engineering field. Lastly, we did not use the term RDD to avoid bias in participants.

A threat to conclusion validity could arise from the exploratory nature of our survey. As such, we did not have formal hypotheses, e.g. for testing if RDD exists, but relied on our interpretation of distributions. To minimize researcher bias, we discussed all important results between the first three authors until consensus was reached. A confounding factor for answers to the *applicant* perspective could be prescribed technology choices by employers, as reported by 43% of our participants. We may have missed other such factors for the phenomenon, which is also supported by the low degree of variance our regression models are able to explain. Moreover, while using linear regression instead of simple correlation analysis improved our understanding of relationships in our data, we still cannot make statements about causality.

For scientific SE surveys, we had a high number of 558 responses for the *applicant* and 130 for the *hiring* perspective, with diversity in work experience, company size, and domain. A threat to external validity, i.e. generalizability, may be that the majority of participants were located in Germany (~90%). Regional and cultural factors could influence the phenomenon and results could partially differ in a sample dominated by participants from, e.g., the US. Furthermore, our *applicant* sample contained 102 students (18%). While we think that the views of students are also relevant for the *applicant* perspective, they may differ from the opinions of professionals. Nonetheless, we still believe the fundamentals of our theory to be applicable to a broad range of software engineering contexts.

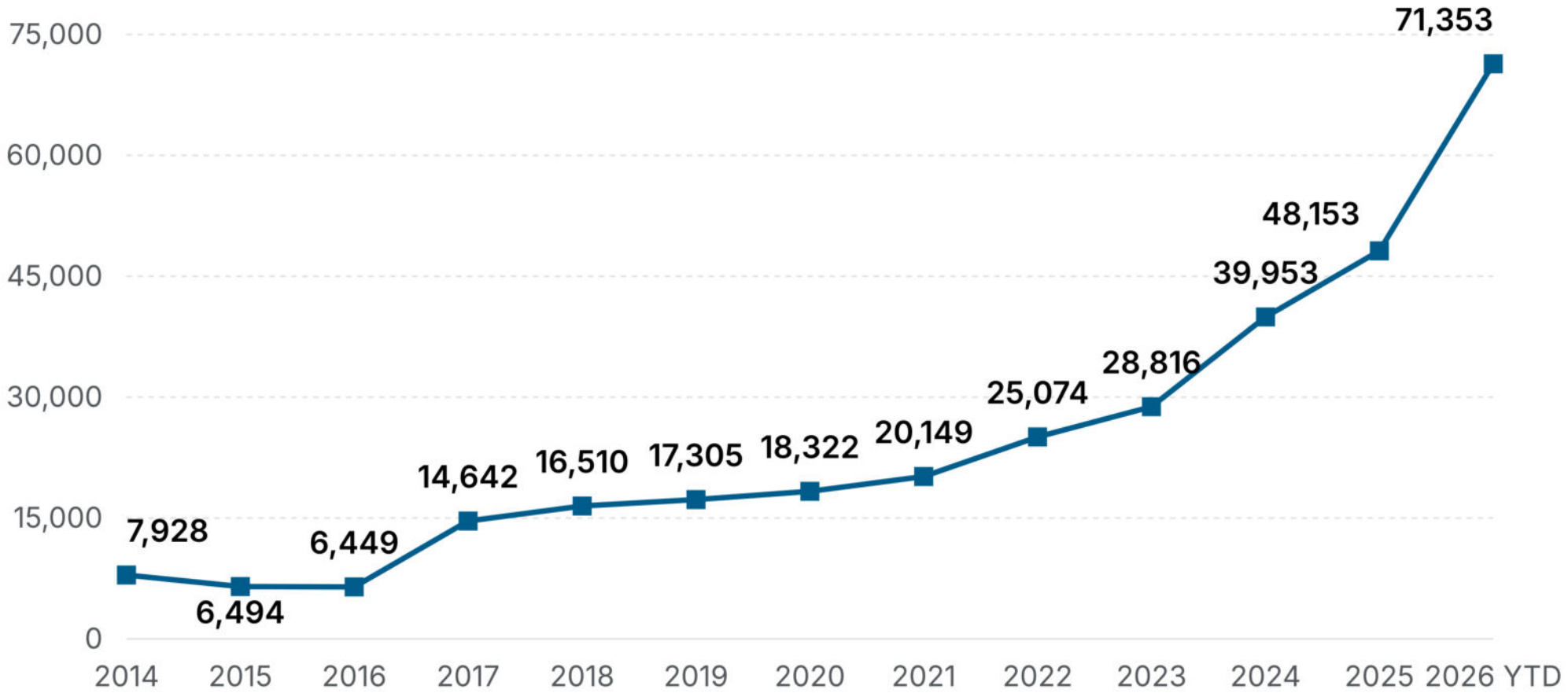


Sources: "Resume-Driven Development: A Definition and Empirical Characterization," Jonas Fritzsch, Marvin Wyrich, Justus Bogner, Stefan Wagner, January, 2021.

#8
Keeping up with security & compliance

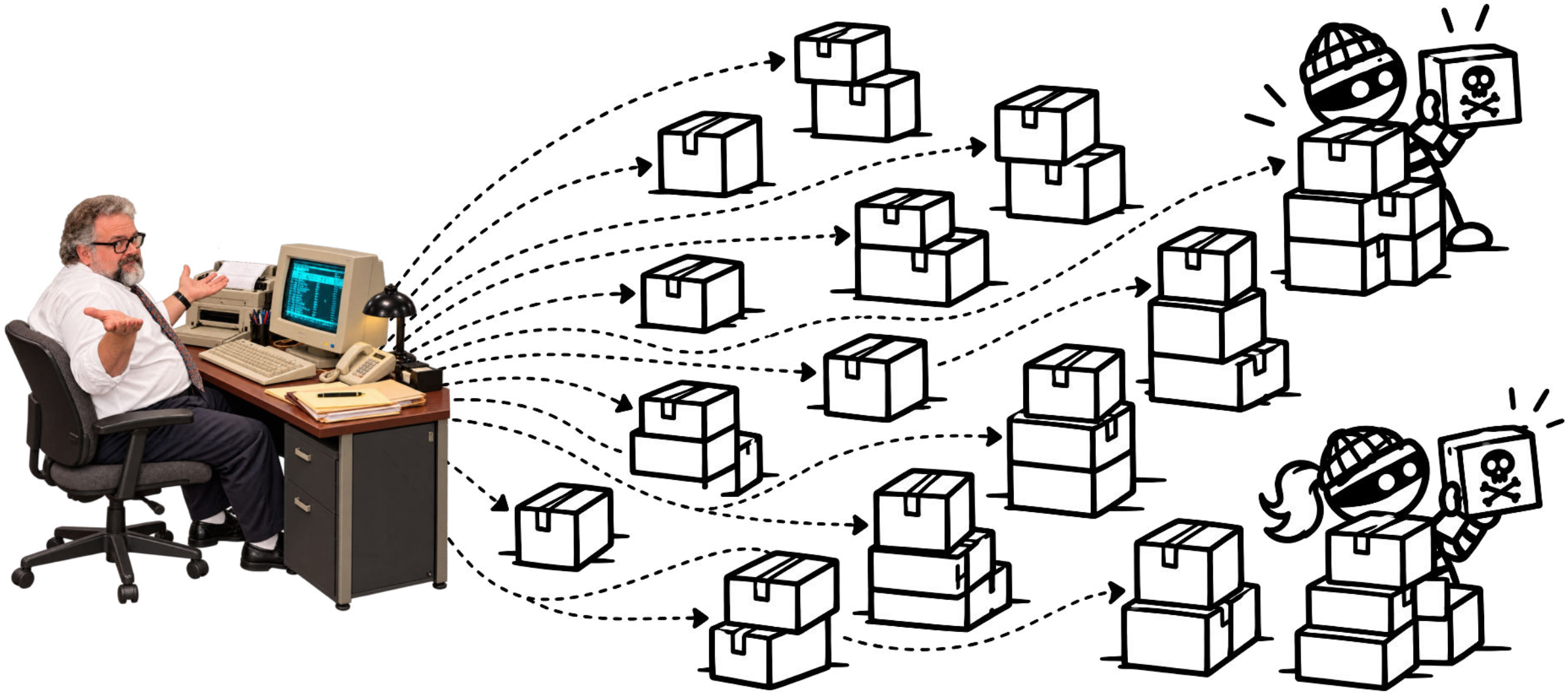
CVEs reported by year

CVE records per calendar year, 2014 to 2026. 2026 is still running.



2026 has nine months on the board, not twelve - and has already beaten all of 2025 by half again.

Source: CVE.ICU, cve.icu, data as of 2026-09-28.



“Don’t build something if you can buy it.”

Sarah Wells, formally at FT, August, 2024.

“Do not blindly start with Kubernetes. Seriously. If your application can get by with a simple PaaS or Serverless offering I’d consider that first. Even VMs make sense for most situations.”

Kelsey Hightower, Autum, 2025

“It’s not about rebuilding what we can purchase that is available on the market. It’s about making sure we spend our time building the things that are bespoke and important for our organization.

Abby Bangser, Syntaso, November 2025.

“Buy everything you can.”

Abby Bangser, Syntaso, November, 2025.

Stop building your own platforms.

Start building your own apps.

Thanks!

https://talks.cote.io/diy/

michael.cote@broadcom.com



Slides & stuff