Platform Engineering Strategy That Sticks

From Hype to Organizational Reality

Faisal Afzal | Principal Technical Consultant, AHEAD | Platform Engineering & CNCF Ambassador

SCaLE 23x MAR 5-8, 2026

Agenda

1. The Problem with the Hype
2. Why Platform Engineering, Why Now
3. Gartner Hype Cycle
4. Defining the Thing: PE, IDP & DevEx
5. What a Mature IDP Looks Like
6. Why Platforms Matter?
7. The Anti-Pattern: Don't Start with the Portal
8. Platform Maturity Assessment
9. Team Structure Importance
10. Platform Team Personas & Stakeholders
11. Common Team Anti-Patterns to Avoid
12. Making the Business Case: Platform ROI
13. Measuring What Actually Matters: Beyond DORA
14. Platform Strategy That Sticks: Your Roadmap
15. Key Takeaways + Q&A

The Problem with the Hype

Standing up a portal and calling it done is one of the most common and costly mistakes teams make.

- Platform Engineering is Gartner's top emerging tech trend with its own dedicated Hype Cycle
- Most teams hear ”platform engineering” and immediately spin up a Backstage portal
- The result: low adoption, frustrated developers, and a platform that quietly dies after 12 months
- The real issue is not tooling but the absence of strategy, team design, and product thinking
- This session gives you the framework to build a platform that lasts

Why Platform Engineering, Why Now

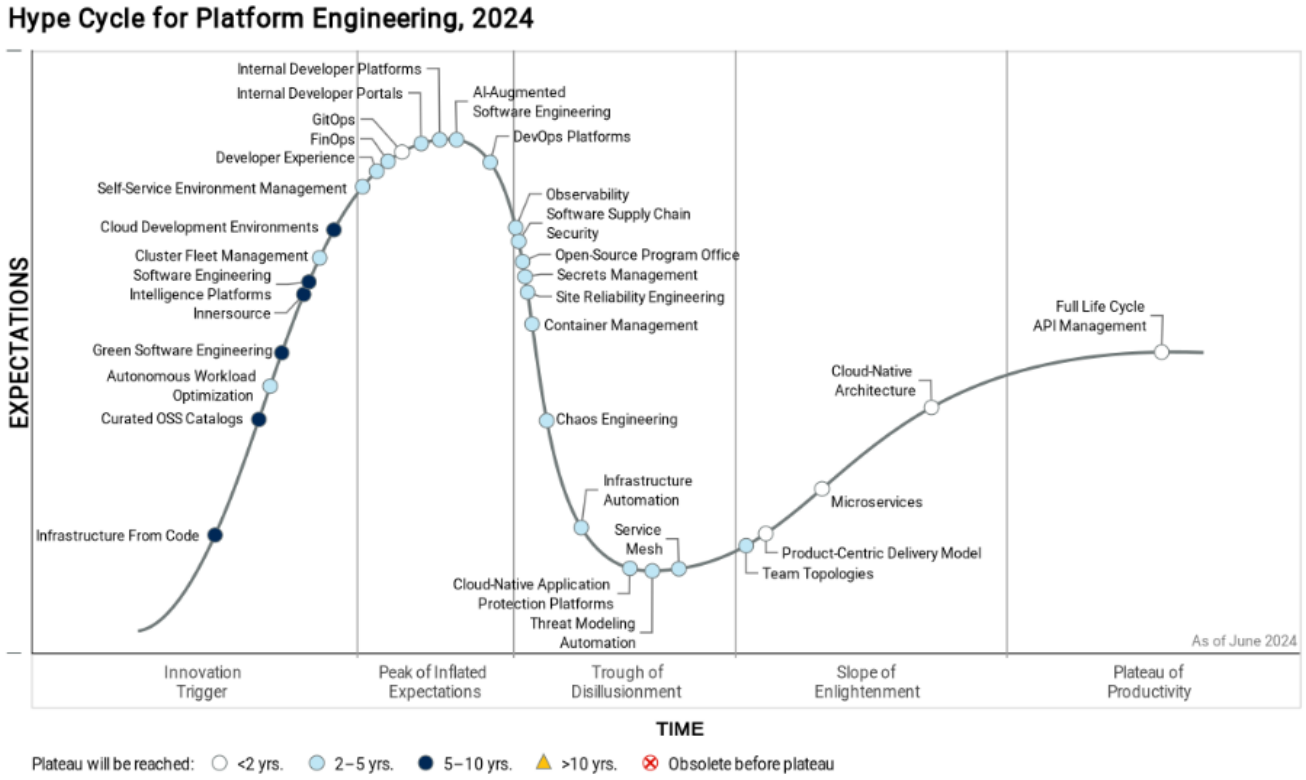
Industry Momentum

- PlatformCon didn't exist before 2021
- Platform Engineering community: 21K+ members on Slack/Discord
- Gartner elevated Platform Engineering to its own Hype Cycle (previously nested under Software Engineering)
- Now mentioned across 7 Gartner Hype Cycles: SRE, Container Technology, Infrastructure Platforms, and more
- Ecosystem explosion: new CNCF projects, tooling, and frameworks every quarter

The Forcing Function: Cloud Entropy

- Cloud native adoption created an explosion of tooling with the CNCF landscape alone has 1,000+ projects
- DevOps said “you build it, you run it.” Reality: scaling that expectation to 500 developers broke down
- Junior developers can't be expected to understand a cloud footprint built for distributed systems
- Entropy: without structure, inconsistency and manual drift are inevitable
- Platform Engineering is the structured, product-minded response to that entropy

Gartner Hype Cycle



Gartner

Defining the Thing: PE, IDP & Developer Experience

Platform Engineering

- The discipline of designing and building toolchains and workflows that enable self-service capabilities for software engineering organizations in the cloud-native era.
- If you are helping developers ship software with less friction, you are doing PE.

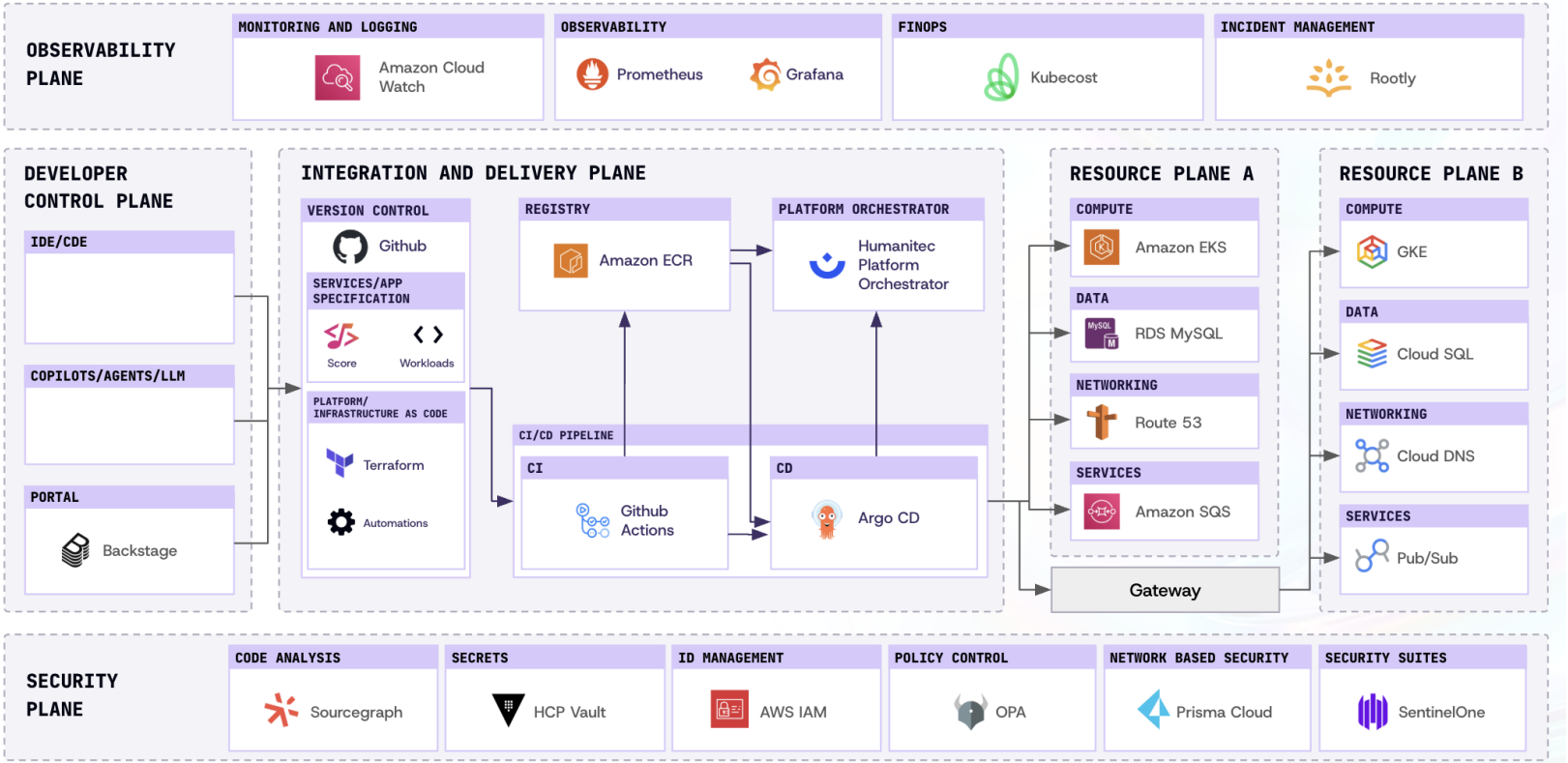
Internal Developer Platform (IDP)

- A self-service layer of APIs, tools, services, and golden paths that allows app teams to deliver at higher pace with reduced coordination.
- It abstracts infrastructure complexity and lowers cognitive load. An IDP is not the same as a portal.
- The portal is just the front door.

Developer Experience (DevEx) as North Star

- The sum of tools, workflows, environments, and culture that affects developer productivity and satisfaction.
- Evolving from 'developer happiness' to a business-aligned performance metric.
- Strong DevEx = better feature velocity, lower attrition, higher quality.

What a Mature IDP Looks Like



Source: Platform Engineering

Why Platforms Matter?

Pain

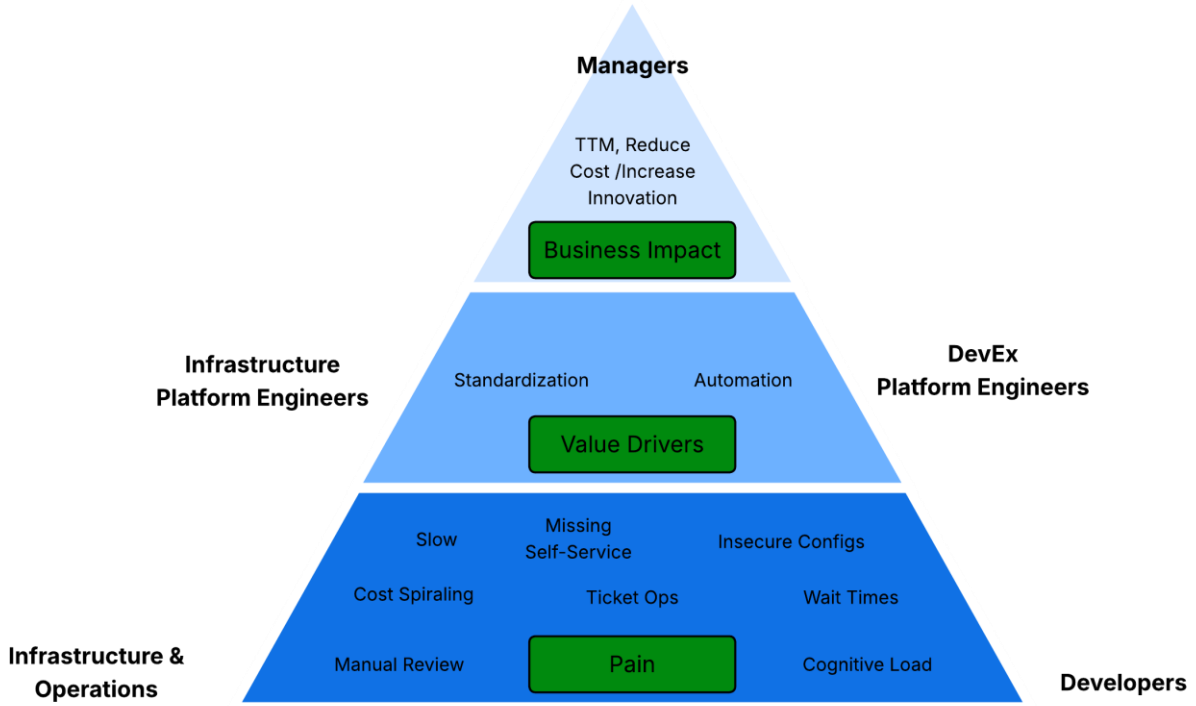
- Slow self-service and ticket-ops
- Manual review & cognitive workload
- Insecure configs & cost spiraling

Value Drivers

- Standardization across teams
- Automation of repetitive workflows
- Golden-paths & self-service tooling

Business Impact

- Faster Time to Market (TTM)
- Reduced infrastructure cost
- Increased innovation capacity



Source: [Platform Engineering]

The Anti-Pattern: Don't Start with the Portal

> *"Internal Developer Portals serve as the interface through which developers can discover and access platform capabilities." — Gartner*

The House vs. Door Mental Model

- Teams that start with Backstage tend to have very low adoption after 12 months
- A portal is the door. Do you build a house by starting with the door or foundation?
- Real value lives is in the orchestration engine, golden paths, automation, environment management
- Portals expose your platform's logic. If the back end is not solid, the portal just exposes the gaps

Build in this order:

Platform Maturity Assessment

Dimension	Provisional	Operational	Scalable	Optimizing
Investment	Temporary / Voluntary	Dedicated Team	Treated as Product	Enabled Ecosystem
Adoption	Erratic	Extrinsic Push	Intrinsic Pull	Participatory
Interfaces	Custom Processes	Standard Tooling	Self-Service Solutions	Integrated Services
Operations	By Request	Centrally Tracked	Centrally Enabled	Managed Services
Measurement	Ad Hoc	Consistent Collection	Insights Driven	Quantitative + Qualitative

Source: Platform Engineering

Team Structure Importance

Conway's Law in Practice

"Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure." — Mel Conway

Team Topologies: The Design Framework

Stream-Aligned

Aligned to a business domain. Delivers product features. Consumes platform capabilities. This is your app/product team.

Platform Team

Provides internal capabilities as a service: CI/CD, infra modules, golden paths, observability. Treats platform as a product.

Enabling Team

Ephemeral. Coaches other teams to adopt new tools and practices (e.g., GitOps, observability). Facilitates, doesn't deliver for them.

Complicated Subsystem

Deep specialization required (e.g., ML infra, low-level networking). Not where you start. Operates in X-as-a-Service mode.

Source: Team Topologies

Platform Team Personas & Stakeholders

Core Platform Roles

Head of PE:

Sets vision, strategy, success metrics. Champions Platform-as-Product mindset.

Infrastructure Platform Engineer:

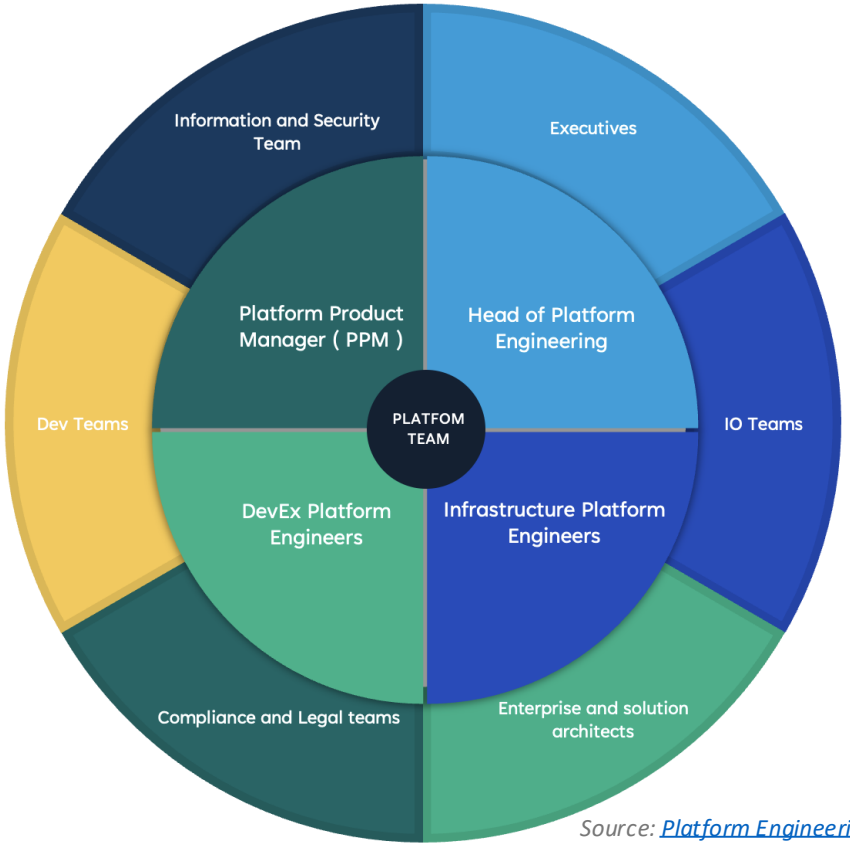
Builds/maintains K8s, networking, IAM. Owns foundational components.

DevEx Platform Engineer:

Bridge between devs and PE. Owns golden paths, UX, feedback loops, adoption.

Platform Product Manager:

Owns roadmap and lifecycle. Aligns developer needs with business goals.



Source: Platform Engineering

Rule of thumb: 1 platform engineer per 20 developers. Adjust based on IDP complexity, regulated industry, and polyglot architecture.

Common Team Anti-Patterns to Avoid

Poor Documentation & Undefined APIs

Problem: Developers lose confidence, build workarounds, duplicate functionality. Trust in the platform erodes quickly.

Remedy: Define service contracts explicitly. APIs, expectations, and usage boundaries must be documented and version-controlled.

'Temporary' Collaborations That Drag On

Problem: Enabling teams spin up short-term partnerships that extend indefinitely, blurring ownership and accountability.

Remedy: Set explicit scopes, timeboxes, and exit criteria for every cross-team engagement.

Over-Functioning Enabling Teams

Problem: Enabling teams become delivery resources instead of coaches. App teams never build independent capability.

Remedy: Refocus enabling teams on coaching and unblocking — not delivering on behalf of others.

Making the Business Case: Platform ROI

Translate technical outcomes into financial language. Business stakeholders prioritize data over demos.

Impact Area	Scenario Example	Monetization Approach
Developer Productivity	2 hrs/week saved × 150 devs	Hourly rate × 2 × 150 × 52 = annual savings
Accelerated Delivery	2–3 more features/month	Estimate revenue contribution per feature × 12
Improved Resilience	4 fewer downtime hrs/month	Cost per downtime hour × 4 × 12
Tooling Consolidation	4 redundant SaaS tools removed	Sum of licenses + support + vendor costs
Talent Retention	Improved developer NPS	Estimated backfill cost per engineer × reduced attrition

ROI Formula: (Total Value Generated – Total Cost) ÷ Total Cost. Platform value compounds: Year 1 may show net cost; by Year 3–4, value creation typically outpaces investment by 2×.

Measuring What Actually Matters: Beyond DORA

DORA Metrics Are Important — But They're Lagging Indicators

- DORA reflects outcomes after the fact but doesn't tell you why a metric changed
- DORA does not capture platform team contributions: pipeline improvements, self-service enablement, golden path adoption
- Affected by variables outside platform control: app complexity, feature type, team structure

Complement DORA with Developer Experience Metrics

Productivity & Flow

- Onboarding time (new eng → first PR)
- Environment setup time
- Time to bootstrap new app/resources
- Deployment frequency (self-service path)

Satisfaction & Trust

- Developer NPS (internal)
- Tooling adoption rate
- Qualitative friction surveys
- Cognitive load assessments

Platform Health

- Golden path adoption %
- # of support tickets (decreasing = good)
- Feature adoption over time
- Voluntary vs. mandated usage ratio

Platform Strategy That Sticks: Your Roadmap

Phase 1: Define & Assess

- Identify strategic intent and developer pain points
- Conduct current-state assessment across 5 maturity dimensions
- Map team structure using Team Topologies

Phase 2: Build the Thin Viable Product (TVP)

- Start with the Thinnest Viable Platform
  - Minimum APIs, golden paths, docs
- Prioritize capabilities that serve 80% of use cases first
- Avoid portal-first. Build orchestration and automation first

Phase 3: Measure & Evangelize

- Establish baseline metrics before you build (developer sentiment, onboarding time)
- Build internal champions. Onboard early adopters voluntarily
- Track adoption, NPS, and friction points — not just DORA

Phase 4: Scale & Sustain

- Evolve team interactions (collaboration → X-as-a-Service → facilitation)
- Demonstrate ROI to secure continued investment and headcount
- Treat the platform as a product
  - With a roadmap, backlog, and SLAs

Key Takeaways

01	Strategy over tooling. Platform Engineering is a discipline, not a product you install. Start with problems, not portals.
02	Your team structure is your architecture. Conway's Law is real. Org design and platform design must evolve together.
03	Treat your platform like a product. That means a roadmap, feedback loops, adoption metrics, and a clear value proposition to your developers.
04	Measure what matters. DORA is a starting point, not a destination. Add DevEx signals and business outcomes to make your ROI case.
05	The Thinnest Viable Platform wins. Don't over-engineer at MVP. Build the smallest thing that accelerates your developers, then iterate.

Questions?

Faisal Afzal
Principal Technical Consultant Lead, AHEAD
Platform Engineering & CNCF Ambassador | Cloud Native LA Organizer

Thank you for attending — slides will be made available at the conference resource page.