

Building a Culture of Continuous Improvement: Fostering Feedback & Psychological Safety

Create an environment where failure is embraced as an opportunity for growth.

SAKSHI NASHA
Senior Software Engineer
@ Cohesity



What is Platform Engineering?



Modern organizations (startups to large enterprises) aim to:

- Onboard developers quickly
- 🚀 Deploy code faster
- ⌚ Reduce time to value



Platform engineering focuses on:

- Enhancing productivity
- Accelerating application cycle time
- Increasing speed to market



Key objective:

- Address developer pain points through an Internal Developer Platform (IDP)
- Provide reusable tools and shared capabilities

Platform Engineering: Rising Industry Focus

- 80% of orgs to adopt platform engineering by 2026 (Gartner)
- Focus on Building Internal Developer Platforms (IDPs)
- Drives scale & faster business value across domains (e.g., Salesforce, ServiceNow, Twilio)



The Cultural Challenges in Platform Engineering



- Many of these governance processes start out as manual controls.

Manual steps -> **friction** in SDLC -> **decreases team velocity** -> **increases** developer and operations **team frustration**.

- Other issues being **organizational silos**
- **Lack of trust**
- **Slow iteration cycles**
- **Fear of failure.**
- Ultimately undermining the goals of platform engineering.

What is continuous improvement?







Continuous Improvement is an **ongoing journey** to enhance products, processes, and team performance through **small, incremental changes**.

- Encourages **learning & adaptation**
- Focuses on reducing waste and inefficiencies
- Involves **regular reflection** (e.g., retrospectives, reviews)
- Empowers teams to test, learn, and **evolve over time**.

Why are feedback & psychological safety vital for platform teams?

Feedback and psychological safety are crucial for effective platform teams:

-  **Psychological Safety:**
Team members feel safe to speak up, share ideas, and admit mistakes without fear. 2 Way Comprehension (Not communication)
-  **Feedback Culture:**
Enables continuous learning, faster problem-solving, and course correction
-  Builds **trust**, accountability, and collaboration.
-  Leads to better innovation, fewer silos, and **higher team performance**

Way out !!



1. Feedback Loops

- a. What Are Feedback Loops?
- b. Embedding Feedback in the Platform Lifecycle



2. Empower developers through self-service with guardrails

- a. Start Right, Stay Right — Govern by Design
- b. Templates ensure governance, security, and cost optimization

3. Stakeholder Communication

Encouraging open, constructive feedback from stakeholders

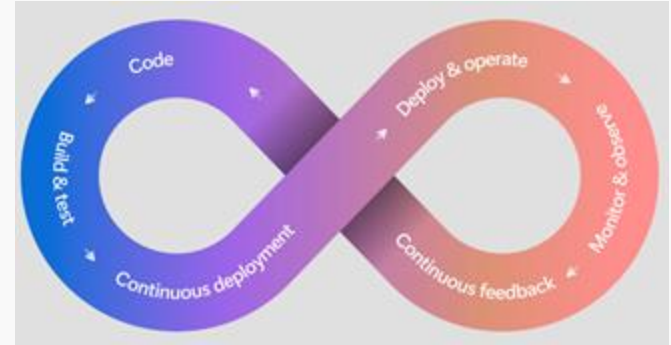
1. Feedback Loops ?

- Feedback loops are continuous cycles of **input** → **action** → **learning** → **improvement**.
- They are essential for evolving platforms based on real user needs.
- Key Types of Feedback:
 -  **Developer Feedback**: Direct input from devs on tool usability, pain points, or friction in workflows.
 - Example: Surveys, slack threads, platform office hours.
 -  **Platform Telemetry**: Data from usage metrics, error rates, and performance logs.
 - Example: Monitoring IDP adoption, tracking CI/CD pipeline times.
- Mindset Tip: **Platform engineering is a journey**—not a big bang. Focus on iterative progress not just top-down driven effort.

B) Embedding Feedback Loops in the Platform Lifecycle

Feedback should be integrated across every stage:





- 🕒 **Planning:** Engage developers **early** to understand **needs** and define priorities
- ✂ **Development:** Gather ongoing input through **demos**, **previews**, and usability tests
- 🚀 **Launch:** Monitor adoption, **collect feedback**, and address **initial friction**
- 🔄 **Post-Launch:** Use **telemetry** and developer feedback to drive continuous refinement



✂ No two companies are the same, so consider the specific needs of your internal customers to plot an incremental course and evolve through this journey.




2. Empower developers through self-service with guardrails


Main Goal : Reduce developer toil while maintaining control

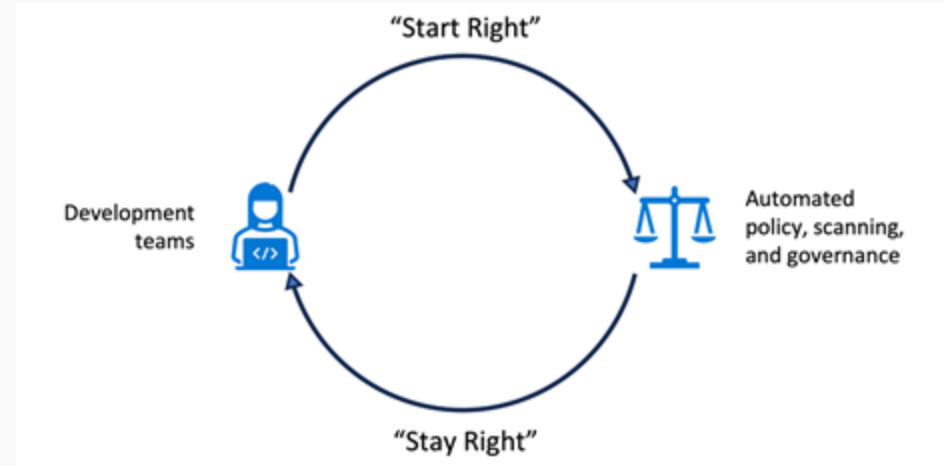
-  **Automation & Policy**
Ensure security, compliance, operational standards, and cost control
-  **Paved Paths with Minimal Learning Curve**
Self-service experiences simplify common tasks and increase developer autonomy
-  **Key Enablers (CATALYST)**
Infrastructure as Code (IaC), Continuous Delivery (CD) pipelines, and GitOps tools enable safe, repeatable automation
-  **Outcome**
Visibility and consistency across developers, operations, and management

A) Start Right, Stay Right – Govern by Design

Main Goal : Templates + automation enable built-in governance




-  **Start Right**
Templates guide developers from day one with secure, compliant foundations
-  **Stay Right via Automation**
Ongoing policy enforcement, scanning, and governance embedded in workflows
-  **Feedback Loop**
Development Teams → Stay Right → Automated Governance → Back to Teams

 Create a system where developers don't have to slow down to stay secure and compliant



B) Templates ensure governance, security, and cost optimization

Main Goal : Templates are more than a starting point—they sustain best practices

-  **Beyond Bootstrapping**
Templates should **enforce policies, enable security scanning**, and support governance throughout the lifecycle
-  **Cost Optimization**
Standardized templates help **reduce redundant spend** across tools, vendors, and cloud resources
-  **Best Practice by Default**
Templates encode **proven configurations** to **reduce risk and manual test effort**

Templates are a strategic lever for sustainable, secure, and cost-effective development

3. Stakeholder Communication



Key Message: Build a **customer-centric platform** by listening and evolving


🕒 Every organization is **different** — tailor your approach to internal customer needs


👤 **Recognize all** stakeholders as valued customers


⚙️ Empower **developers** through customer-centric internal platforms


❓ Ask Yourself : *Will this feel like an improvement or just another tool or new just another new User Experience?*

Tips for getting useful, honest feedback from developers and users

 **Psychological Safety:** People feel safe to speak up without fear of judgment or punishment

 **Create a Safe Environment:** Normalize sharing feedback, asking questions, and reporting failures

 **Failure-Positive Culture:** View mistakes as opportunities to learn, not blame

 Ask open questions, listen actively, and act on what you see (rather than hear)

Honest feedback fuels platform improvement and adoption

Thank you

Connect with me 

