Reorganizing Hybrid Infrastructure Engineering for a Product-Centric Future: A Strategic Blueprint

1. Executive Summary

This report outlines a strategic transformation for the Fortune 100 company's hybrid infrastructure engineering organization. The current structure, characterized by traditional technology silos, has resulted in operational inefficiencies, including poor collaboration, knowledge gaps, and an imbalanced distribution of resources. The proposed reorganization aims to transition this 24-member team into an agile, engineering-focused IT infrastructure shared services function. This new model will pivot away from a predominantly operational focus—while retaining essential Tier-3 support capabilities—to emphasize 'IT as a Service' (ITaaS). Core to this transformation is the delivery of user-consumable infrastructure products, enabled by self-service automation, a pervasive product mindset, comprehensive full lifecycle management of these products, and robust, built-in governance.

Key recommendations center on adopting platform engineering principles to create a foundation for these infrastructure products. Teams will be restructured around these clearly defined products, fostering a product-centric culture and operational approach. Crucially, a structured and empathetic change management process will be vital to navigate this transition successfully, particularly in integrating all team members and addressing skill evolution.

The anticipated benefits of this reorganization are substantial. They include significantly improved collaboration across formerly siloed disciplines, accelerated delivery of infrastructure services to internal consumers, and consequently, enhanced satisfaction among internal users such as development teams and business units. Furthermore, this model promises more effective and integrated governance, increased operational efficiency, and a marked improvement in the infrastructure organization's agility to respond to and enable the dynamic demands of the business. This strategic shift will position the infrastructure team as a key enabler of business value within the enterprise.

2. Pivoting to a Product-Oriented Infrastructure Operating Model

The evolution of IT infrastructure management within large enterprises necessitates a fundamental shift in perspective and operational strategy. Moving beyond traditional, technology-centric silos towards a model where infrastructure is conceived, delivered, and managed as a portfolio of user-consumable products is no longer a futuristic

ideal but a contemporary imperative. This transition, underpinned by a robust product mindset and enabled by disciplines like platform engineering, is critical for enhancing agility, efficiency, and business alignment.

The Imperative of 'IT as a Service' and a Product Mindset for Modern Infrastructure

Defining Infrastructure as User-Consumable Products

The core of this transformation lies in redefining infrastructure. Instead of viewing it as a collection of disparate technologies (servers, storage, networks) managed by specialized, often isolated, teams, the organization will transition to offering a portfolio of standardized, well-defined services or "products".[1] These products are designed with the internal user—be it a developer, a data scientist, or a business unit—as the primary customer. This approach involves a shift from managing technology components to delivering holistic solutions that provide clear value.[2]

Each infrastructure product must have clearly articulated service definitions, standardized interfaces for consumption (such as APIs or a self-service portal), and formally defined Service Level Agreements (SLAs) that set expectations for performance, availability, and support.[3] Furthermore, these products will be subject to rigorous full lifecycle management, from conception and development through to operation and eventual retirement.

Adopting an "IT as a Service" (ITaaS) model is central to this vision. In this paradigm, the infrastructure group operates akin to an internal service provider, deeply attuned to the needs of its customers and focused on delivering tangible value.[4] This is a significant departure from a purely operational or "keep the lights on" mandate. It necessitates understanding the "jobs to be done" for internal users and designing infrastructure services that directly enable their success. IaaS (Infrastructure as a Service) concepts, which emphasize on-demand access to computing resources, provide a useful, albeit external, parallel for how internal services can be structured and consumed.[6]

Crucially, this operational shift must be accompanied by a cultural one: the adoption of a "product mindset".[8] This means applying principles and practices from software product management to the domain of infrastructure services. It involves deeply understanding customer (internal user) needs and pain points, defining a clear product vision and strategic roadmap for each infrastructure offering, actively seeking and incorporating user feedback, and iterating on products to continuously enhance their value and usability.[8] The infrastructure team, therefore, moves from being

reactive order-takers to proactive owners and innovators of their service portfolio. This redefinition of the IT-business relationship is profound. It transforms the infrastructure team from a perceived cost center, often seen as a bottleneck, into a strategic partner that actively enables business agility and innovation. This requires the team to cultivate not only technical excellence but also business acumen, strong communication skills, and a relentless customer-centric focus, much like successful external product organizations.[2] Success is no longer measured solely by uptime or ticket closure rates but by the value delivered, user satisfaction, and the adoption rate of these internal products.[8]

Benefits for a Fortune 100 Enterprise

For a Fortune 100 company, the transition to a product-oriented infrastructure model offers compelling strategic advantages:

- **Increased Business Agility:** By providing standardized, readily available, and self-service infrastructure products, the organization can respond much more rapidly to evolving market conditions and new business demands. Development teams can access the resources they need quickly, accelerating project timelines and time-to-market for new business initiatives.[8]
- **Improved Efficiency and Cost Optimization:** Standardization of service offerings, coupled with extensive automation and self-service capabilities, significantly reduces manual effort, minimizes human error, and lowers operational overhead.[8] This frees up skilled engineers to focus on higher-value work, such as product innovation and strategic improvements.
- **Enhanced Developer Productivity:** Providing developers with on-demand, self-service access to infrastructure resources (compute, storage, networking, development tools) dramatically reduces traditional wait times and handoffs. This allows development teams to focus their energy and expertise on building and iterating on business applications, rather than grappling with infrastructure procurement and configuration.[14]
- **Better Alignment with Business Outcomes:** When infrastructure teams adopt a product mindset, their primary focus shifts to delivering value to their internal customers. Since these internal customers (e.g., application development teams) are themselves working towards business objectives, the infrastructure products inherently become more closely aligned with and supportive of broader business goals.[8]
- **Improved Governance and Compliance:** Integrating governance and security controls directly into the design, automation, and delivery processes of infrastructure products ensures consistency, enhances security posture, and

reduces compliance-related risks.[15] This "governance by design" approach is particularly critical in the complex regulatory environment of a Fortune 100 company.

Case studies from organizations like Salesforce, which underwent a similar transformation in its Business Technology group, demonstrate these benefits vividly, including reduced time to market, improved IT-business collaboration, and enhanced operational efficiency.[2] Other large enterprises also report significant gains from adopting product-centric IT models.[11] Industry analysts like Forrester and Gartner consistently advocate for this shift as a key enabler of digital transformation and business success.[17]

Platform Engineering: The Engine for Your Infrastructure Product Strategy

Platform engineering emerges as the critical discipline to realize the vision of infrastructure as a product. It provides the methodologies, technical capabilities, and team focus required to build and operate the foundation upon which these user-consumable infrastructure services will be delivered.

Core Concepts and Responsibilities

At its heart, platform engineering is about designing, building, and maintaining an **Internal Developer Platform (IDP)**.[14] An IDP is not a single tool but rather a curated ecosystem of tools, standardized services, automated workflows, and "golden paths" that collectively enable developers and other internal users to self-serve their infrastructure and operational needs efficiently and safely.[19] The platform engineering team treats this IDP as an internal product, continuously evolving it based on user feedback and changing technological landscapes, with a keen focus on developer experience, reliability, and scalability.[14]

The responsibilities of a platform engineering team are comprehensive and span several key areas [22]:

- **Infrastructure as Code (IaC):** Championing and implementing IaC practices (using tools like Terraform, Ansible) to ensure that all infrastructure components are defined, provisioned, and managed through code, enabling consistency, repeatability, and version control. [23]
- **CI/CD for Infrastructure:** Building and maintaining robust Continuous Integration and Continuous Delivery (CI/CD) pipelines specifically for infrastructure changes, allowing for automated testing and deployment of infrastructure products. [23]
- **Container Orchestration:** Managing and evolving container platforms (e.g., Kubernetes) if they are part of the technology stack, providing a standardized

environment for deploying and scaling containerized applications. [23]

- **Observability:** Implementing comprehensive monitoring, logging, tracing, and alerting solutions to provide deep visibility into the health and performance of the platform and the infrastructure products it delivers. [22]
- **Security Integration:** Embedding security best practices, tools, and automated checks throughout the platform and the lifecycle of infrastructure products (DevSecOps). [22]
- **Developer Experience (DX):** Continuously seeking to improve the ease of use, efficiency, and satisfaction of developers interacting with the platform and its services. [22]

While platform engineering provides the technical "how-to" for building self-service infrastructure, it is the product mindset that guides the "why" and "what." The platform itself, and all the services it exposes, must be conceived, designed, and iterated upon as products. This means understanding the specific problems the platform is solving for its users (primarily developers), defining clear value propositions, gathering feedback, and prioritizing features based on user impact.[15] Without this product-centric approach, an IDP risks becoming merely a collection of sophisticated tools that fail to address real user needs or gain widespread adoption.

How Platform Teams Enable Self-Service and Governance

Platform teams are pivotal in delivering the dual mandate of self-service automation and built-in governance:

- **Self-Service Enablement:** They achieve this by creating and curating a catalog of standardized templates, "golden path" blueprints (pre-approved, secure, and efficient ways to accomplish common tasks), and fully automated workflows.[19] Developers and other users can then consume these resources on-demand through the IDP (e.g., a web portal, CLI tools, or API integrations) to provision and manage infrastructure without requiring manual intervention from the platform team.[19]
- **Embedded Governance:** Governance is not an afterthought or a separate manual process but is woven directly into the fabric of the platform and its automated workflows.[19] This is realized through:
    - **Policy as Code (PaC):** Defining security, compliance, and operational policies in a machine-readable format (e.g., using Open Policy Agent - OPA) and automatically enforcing them during provisioning and configuration.[19]
    - **Automated Compliance Checks:** Integrating tools that continuously monitor configurations against defined standards and automatically report or remediate deviations. [14]

- **Security Scanning:** Embedding vulnerability scanning, static code analysis for IaC, and other security checks directly into CI/CD pipelines for infrastructure. [14]
- **Role-Based Access Control (RBAC):** Implementing fine-grained access controls within the IDP to ensure users can only access and operate on resources appropriate to their roles and responsibilities. [23]

This approach of embedding governance allows for a crucial balance: developers gain the speed and autonomy they need, while the organization maintains the necessary control and risk management posture essential for a Fortune 100 enterprise.[16] It transforms governance from a potential bottleneck into an enabler of safe, rapid innovation. The "right way" to consume infrastructure becomes the "easy way" because compliant and secure paths are pre-engineered into the self-service offerings.[25] This requires a significant upfront investment in defining, codifying, and automating these policies, but the long-term benefits in terms of speed, consistency, and reduced risk are substantial.

Integrating DevOps and SRE Principles for Enhanced Agility and Reliability

To fully realize the benefits of a product-oriented infrastructure model, the principles of DevOps and Site Reliability Engineering (SRE) must be deeply integrated into the new teams' culture and practices.

Breaking Down Silos and Fostering Collaboration

The current challenges of poor collaboration and knowledge gaps are directly addressed by the core tenets of DevOps: enhanced communication, close collaboration between all stakeholders (development, operations, security, etc.), integration of processes and tools, and extensive automation.[4] By structuring teams around end-to-end ownership of infrastructure products, cross-functional collaboration becomes a natural mode of operation rather than an exception.[27] These product teams will inherently embody DevOps practices, sharing responsibility for the entire lifecycle of their services, from design and development through deployment and ongoing operations. [4]

Ensuring Operational Excellence Alongside Product Development

While DevOps fosters agility and speed in development, Site Reliability Engineering (SRE) brings a rigorous, engineering-driven approach to operational excellence and reliability.[19] Integrating SRE principles is crucial for ensuring that the infrastructure products are not only innovative and feature-rich but also exceptionally reliable,

scalable, and maintainable. Key SRE practices to be adopted include:

- **Defining and Adhering to Service Level Objectives (SLOs):** Each infrastructure product will have clearly defined SLOs for availability, performance, and other critical metrics, agreed upon with its consumers. These SLOs will guide engineering efforts and operational priorities.[30]
- **Error Budgets:** Derived from SLOs, error budgets provide a quantitative measure of allowable unavailability or performance degradation. They empower teams to balance the risk of innovation with the need for reliability, making data-driven decisions about when to focus on new features versus stability improvements. [30]
- **Reducing Toil:** A relentless focus on identifying and automating manual, repetitive, and automatable operational tasks (toil) to free up engineering time for more valuable, proactive work. [30]
- **Effective Incident Management:** Implementing robust incident response protocols, including clear escalation paths, on-call rotations within product teams, and blameless post-incident reviews aimed at learning and systemic improvement. [30]

By embedding SREs or, more broadly, SRE practices and mindset within each infrastructure product team, the organization ensures that operational excellence and reliability are shared responsibilities, directly influencing product design, development, and ongoing management.[27] This is particularly vital for effectively managing the Tier-3 support responsibilities that will remain within this engineering-focused organization.

3. Designing Your New Product-Oriented Team Structures

The transition to a product-oriented operating model requires a thoughtful redesign of team structures. This involves defining a clear portfolio of internal infrastructure products and then aligning small, empowered teams to own these products end-to-end, including their delivery across the hybrid cloud landscape.

Defining Your Internal Infrastructure Product Portfolio

The first critical step is to identify and define the core "product families" that the infrastructure organization will offer. This process moves away from thinking about individual technologies (like a specific storage array or hypervisor version) and towards defining user-centric services that deliver clear value to internal consumers.[31]

Identifying Core Infrastructure Product Families

Based on the current technology landscape (Virtualization, Hardware Standards,

Windows, Linux, Active Directory, Storage, Backup, Private Cloud, Public Cloud) and the strategic goals of delivering user-consumable products and enabling self-service, several logical product families can be delineated [31]:

- **Compute Platform Services:** This family would offer a catalog of standardized compute resources. Examples include:
  - *Virtual Machine Services (Linux & Windows):* Pre-defined and custom-configurable VMs with various OS, CPU, and RAM options, abstracting the underlying virtualization (e.g., VMware on-premises) and hardware. [38]
  - *Container Platform Services:* If applicable or planned, a managed Kubernetes service or similar container orchestration platform. [38]
  - *Serverless Function Services:* Potentially, a platform for deploying and running serverless functions. [39]
- **Enterprise Storage Solutions:** This family would provide a portfolio of storage offerings catering to different needs. Examples include:
  - *Block Storage Services:* Tiered block storage for VMs and databases. [38]
  - *File Storage Services:* Enterprise-grade NFS/SMB shares. [38]
  - *Object Storage Services:* Scalable object storage for unstructured data, backups, and archives. [38]
  - *Data Protection Services:* Integrated backup and recovery solutions for all storage offerings. [38]
- **Network Services & Automation:** This family focuses on providing connectivity and network-related functionalities. Examples include:
  - *Virtual Networking Services:* Self-service provisioning of VPCs/VNETs, subnets, and routing. [38]
  - *Load Balancing Services:* Options for application load balancing. [38]
  - *DNS & IPAM Services:* Management of DNS records and IP address allocation. [33]
  - *Firewall & Security Group Management:* Self-service requests for firewall rule changes or security group configurations, governed by policy. [33]
- **Identity & Access Management (IAM) Services:** This family centralizes identity and access controls. Examples include:
  - *Directory Services:* Managed Active Directory services for authentication and authorization. [33]
  - *Privileged Access Management (PAM) Services:* Solutions for managing and securing privileged accounts. [33]
  - *Federated Identity Services:* Enabling single sign-on and consistent identity across on-premises and cloud applications. [33]
- **Automation and Developer Enablement Platform:** This product family provides

a suite of tools and services aimed at enhancing developer productivity, driving automation across infrastructure products, and exploring AI Ops capabilities. Examples include:
    - *CI/CD Pipeline Services:* Standardized, self-service CI/CD pipelines for application and infrastructure code. [38]
    - *IaC Provisioning Services:* A catalog of IaC modules and templates for common infrastructure patterns. [38]
    - *Observability Services:* Centralized logging, monitoring, and alerting tools for applications and infrastructure. [38]
    - *Artifact Repository Services:* Managed repositories for code, container images, and other build artifacts. [38]
    - *Automation Frameworks & Tooling:* Shared libraries and platforms to accelerate automation efforts within other product teams. [15]
    - *AI Ops Exploration & Tools:* Investigation and piloting of AI tools for operational intelligence and automation. [15]
- **Cloud Gateway Services:** This family focuses on governing and facilitating the use of public cloud resources. Examples include:
    - *Cloud Account Provisioning & Governance:* Automated setup of public cloud accounts/subscriptions with baseline security and compliance configurations. [42]
    - *Cloud Cost Management & Optimization Services:* Tools and guidance for tracking, analyzing, and optimizing public cloud spend. [42]
    - *Secure Cloud Connectivity Services:* Standardized solutions for connecting on-premises environments to public clouds (e.g., VPN, Direct Connect/ExpressRoute). [42]

The definition of these product families must be an iterative process, driven by a deep understanding of the needs, workflows, and pain points of internal users (e.g., application development teams, data analytics groups, business units).[3] The goal is to offer services that are genuinely consumable and provide clear value. This initial product definition phase is not merely an academic exercise; it is a foundational act of transformation. By shifting the focus from managing technologies to delivering defined products, the organization forces a customer-centric perspective from the outset. This process itself helps to break down existing siloed thinking, as teams must collaborate to define products that often span traditional technological boundaries. For example, a "Virtual Machine Service" product inherently involves compute, storage, networking, and OS considerations, requiring input from experts who previously might have operated in separate teams.

Proposed Table: Sample Infrastructure Product Catalog

To make this portfolio tangible, a service catalog is essential. The following table provides an illustrative structure [31]:

Product Family	Product Name	Description	Key Features/ Services Offered	Target Users/Personas	Access Method	SLA Highlights (Illustrative)
Compute Platform Services	Standard Linux VM Service	On-demand, standardized Linux virtual machines for general application hosting.	CentOS, RHEL, Ubuntu options; Pre-defined CPU/RAM/Disk tiers; Automated patching; Basic monitoring included.	Application Development Teams	Self-Service Portal, API	99.9% Availability; 15-min Provisioning
Compute Platform Services	Managed Kubernetes Cluster Environment	A managed K8s environment for deploying containerized applications.	Up-to-date K8s versions; Integrated networking & storage; Self-healing; Optional monitoring/logging add-ons.	DevOps Teams, App Developers	Self-Service Portal, kubectl via API	99.95% Control Plane Availability
Enterprise Storage Solutions	Enterprise NFS Share Service	Highly available, performant NFS	Tiered performance (Standard,	Application Teams, DBAs	Self-Service Portal	99.99% Data Availability

		shares for application data.	Premium); Snapshot capabilities; Quota management; AD integration for access control.			
Enterprise Storage Solutions	S3-Compatible Object Storage	Scalable, durable object storage for unstructured data, backups, and archives.	Unlimited objects/capacity; Versioning; Lifecycle policies; Public/private access options.	Developers, Data Engineers	API, CLI, Self-Service Portal	99.999999999% Durability
Network Services & Automation	Self-Service Firewall Rule Request	Automated workflow for requesting and implementing firewall rule changes.	Policy-driven approvals; Audit trail; Standard change windows; Pre-validation checks.	App Owners, Security Teams	Self-Service Portal	24-hour Turnaround for Standard Requests
Automation and Developer Enablement Platform	CI/CD Pipeline as a Service	Standardized, reusable CI/CD pipelines for common application stacks.	Jenkins/GitLab CI templates; Integrated security scanning; Automated deployment to staging/prod;	Development Teams	Git commit trigger, Portal UI	Pipeline execution success >98%

			Notifications.			
Automation and Developer Enablement Platform	Centralized Observability Service	Unified platform for logging, monitoring, and alerting for applications and infrastructure.	Pre-configured dashboards; Customizable alerts; Log aggregation and search; Distributed tracing.	DevOps Teams, SREs, Product Teams	Self-Service Portal, API	99.9% Platform Availability
Cloud Gateway Services	Governed AWS Account Provisioning	Automated provisioning of new AWS accounts with baseline security & billing.	IAM roles pre-configured; Guardrails for service usage; Cost allocation tags enforced; Centralized logging enabled.	Project Teams, Cloud Architects	Self-Service Portal	1-hour Account Setup

Value of this Catalog: This catalog is not just documentation; it's a cornerstone of the product-oriented model. It provides:

1. **Clarity:** For consumers, it clearly defines what services are available, their capabilities, and how to access them.[31]
2. **Standardization:** It promotes the use of standardized offerings, reducing complexity and bespoke solutions. [31]
3. **Accountability:** It forms the basis for SLAs and helps product teams understand their delivery commitments. [31]
4. **Foundation for Self-Service:** It is the "menu" from which users will select services via the IDP. [31] It directly addresses the user's request for "user-consumable products" and provides a tangible framework for the entire

reorganization.[44]

Organizational Models for Infrastructure Product Teams

With a defined product portfolio, the next step is to structure teams to own and deliver these products.

Exploring Structures

Several organizational models can be adapted for infrastructure product teams:

- **Platform Teams:** This is a common and highly relevant model where one or more central teams are responsible for building, maintaining, and evolving the IDP and the core infrastructure products offered through it.[31] This structure excels at delivering standardized, highly automated, self-service capabilities across the organization and aligns well with the platform engineering discipline.[21]
- **Stream-Aligned Teams (adapted for Infrastructure):** Inspired by Team Topologies, these teams would align to specific internal customer value streams or "journeys".[31] For instance, an "Application Deployment Value Stream Team" might own all infrastructure products (compute, basic storage, CI/CD integration points) necessary to take an application from code to production smoothly. This promotes a very strong customer focus.
- **Squad Model (adapted for Infrastructure):** This involves small (fitting the <8 person rule), cross-functional, and highly autonomous teams, each taking full ownership of a specific infrastructure product or a significant, well-defined component of it.[12] Examples could include a "Database-as-a-Service Product Squad" or a "Network Automation Squad."

The guiding principle, regardless of the specific label, is to create teams with clear, end-to-end responsibility for the design, development, operation, and continuous improvement of their defined infrastructure products.[8]

Aligning Teams to Product Families, Ensuring Team Sizes of 8 or Less

Once infrastructure product families are defined (as per the catalog), teams should be mapped to these families. A single team might own one product family or a few closely related, smaller products. The strict constraint of **8 people or less per team** is a critical design driver. This size promotes agility, enhances communication within the team, and fosters clear ownership and accountability, as requested.

This team size constraint is not merely an administrative preference; it acts as a deliberate design catalyst. It forces a more granular definition of products and product components. If a product family like "Compute Services" is too extensive for

an 8-person team to manage effectively in a Fortune 100 context, it must be decomposed into smaller, more focused product components (e.g., "Linux VM Service Team," "Windows VM Service Team," "Container Platform Foundation Team"). Each of these smaller components can then be owned by a dedicated small team. This inherent modularity can significantly improve agility, focus, and the speed of iteration for each individual team.

However, this modularity also increases the number of inter-team dependencies and elevates the importance of well-defined APIs or service contracts between these component teams. An overarching "platform-of-platforms" or a sophisticated orchestration strategy becomes essential to ensure these modular products work together seamlessly to deliver cohesive value to end-users. Consequently, formal mechanisms for cross-team alignment become even more critical. "Chapters" (communities of practice for specific skills, e.g., all Linux SMEs across different product teams collaborating on OS hardening standards) and "Guilds" (cross-functional groups addressing cross-cutting concerns, e.g., a security guild ensuring consistent security practices across all infrastructure products) are vital for maintaining coherence, sharing knowledge, and driving consistent standards across the product-oriented organization.[18]

It is vital that this new structure does not inadvertently create new silos along product lines. While teams have focused ownership, a culture of collaboration and established processes for managing dependencies are paramount. [32]

Tailoring Structures for Hybrid Cloud Operations

The existing infrastructure is hybrid, encompassing both on-premises (private cloud, traditional virtualization) and public cloud technologies. The new team structure must effectively manage this complexity. [4]

Integrating On-Premises and Public Cloud Management

A core principle should be that infrastructure product teams own their products across the entire hybrid landscape where applicable. For example, the "Compute Platform Services Team" should be responsible for delivering and managing virtual machine services on both the internal VMware stack and across relevant public cloud providers (e.g., AWS EC2, Azure VMs).[4] This means the team needs skills and responsibilities spanning both environments.

This approach is crucial to avoid the pitfall of creating separate "on-prem teams" and "cloud teams" for the same functional product family (e.g., an "on-prem compute

team" and a "cloud compute team"). Such a division would undermine the goal of breaking down silos and fostering integrated, efficient operations. It would also lead to fragmented user experiences and duplicated effort. [13]

Successfully integrating on-premises and public cloud management within single product teams demands more than simply co-locating specialists with different cloud skills. It necessitates a deliberate strategy to cultivate "hybrid-native" engineers—individuals who can think, design, and operate effectively across these diverse environments. This requires a shift in mindset, where engineers understand the unique strengths, weaknesses, and operational nuances of each platform and can design solutions that leverage them appropriately for the benefit of the product. This is supported by a common set of tools and processes that abstract or bridge the underlying differences, promoting consistency.[4]

Ensuring Seamless Service Delivery Across the Hybrid Landscape

The Internal Developer Platform (IDP) should strive to provide a unified, abstracted interface for consuming infrastructure services, masking the underlying hosting complexities from the end-user wherever possible.[4] For instance, a developer requesting a "standard web server" product might select CPU, RAM, and OS, and the platform orchestration, guided by policy (cost, compliance, performance), would determine the optimal hosting location (on-prem or public cloud) without the developer needing to be an expert in both.

Governance policies (security, compliance, cost), automation scripts (IaC), and operational procedures must be applied consistently across the entire hybrid environment to ensure uniformity, reduce operational friction, and simplify management.[13] This consistency is key to realizing the efficiencies of a product model in a hybrid world.

This presents a significant architectural and operational challenge. It requires robust architectural principles, investment in hybrid-aware tooling (e.g., Azure Arc for managing non-Azure resources [4], Terraform for multi-cloud IaC, Ansible for consistent configuration management), and a strong organizational focus on interoperability and standardization.[4] This has profound implications for the organization's training and recruitment strategies (focusing on cross-cloud skills, adaptability, and systems thinking) and its technology selection process (prioritizing tools with strong hybrid capabilities, open standards, and robust API support).

3.4 Establishing the Automation and Developer Enablement Team

To accelerate the adoption of advanced automation techniques, provide a robust developer enablement platform, and drive AI-driven insights across the infrastructure product landscape, establishing a dedicated **Automation and Developer Enablement Team** is highly valuable. This team consolidates the functions previously envisioned for a separate "Developer Enablement Platform Team" and an "Automation / AI Ops Team," functioning as a unified Center of Excellence (CoE) and an Enabling Team. [45] This approach supports the product-aligned teams rather than creating new operational silos. [31]

Core Functions and Value:

- **Developer Enablement Platform Delivery:** This team will own, develop, and operate the "Automation and Developer Enablement Platform" product family. This includes providing self-service offerings such as:
  - *CI/CD Pipeline Services:* Standardized, self-service CI/CD pipelines for application and infrastructure code. [23]
  - *IaC Provisioning Services:* A catalog of IaC modules and templates for common infrastructure patterns. [23]
  - *Observability Services:* Centralized logging, monitoring, and alerting tools for applications and infrastructure. [22]
  - *Artifact Repository Services:* Managed repositories for code, container images, and other build artifacts. [23] The focus will be on excellent developer experience (DX), ease of use, and robust self-service capabilities for these platform products. [22]
- **Advanced Automation Support & CoE:** The team will develop and disseminate automation best practices, create reusable automation frameworks, libraries (e.g., standardized IaC modules beyond basic provisioning, advanced scripting libraries), and shared automation tools that other product teams can leverage. [14] They will also offer expert consultation and hands-on assistance for particularly complex automation challenges faced by individual product teams, ensuring these teams are empowered to automate their own products with centralized support and expertise.
- **AI Ops Investigation and Implementation:** A key responsibility will be to research, evaluate, pilot, and implement emerging AI Ops technologies and tools relevant to infrastructure management. [15] This includes areas like predictive analytics for capacity planning, anomaly detection for proactive issue resolution, intelligent monitoring, and automated remediation workflows. The team will develop expertise in applying AI to optimize infrastructure operations, improve reliability, and enhance security across all product families.
- **Liaison with DevOps Consumers:** This team will act as a specialized interface

with dedicated DevOps teams (and other sophisticated internal consumers). They will gather requirements for cross-cutting automation needs and platform-wide AI-driven capabilities that span multiple infrastructure products. [21] They will also help consumer teams understand how to best utilize the suite of infrastructure products through automated means, potentially by providing SDKs, standardized APIs, or advanced integration points developed in collaboration with the respective product teams.

Benefits:

- **Unified Expertise:** Concentrates deep skills in platform development, developer experience, advanced automation, and AI Ops, which might be challenging to distribute effectively across all product teams initially. [23]
- **Consistency and Standardization:** Promotes the use of common developer tools, automation platforms, and AI Ops practices across the organization, reducing redundancy and improving interoperability. [31]
- **Focused Innovation Hub:** Dedicates focus to building a superior developer platform and exploring cutting-edge AI technologies for infrastructure, driving innovation that individual product teams might not have the bandwidth for. [15]
- **Efficiency Gains:** Provides a cohesive suite of developer enablement products and reusable automation components, alongside centralized AI Ops services, helping all product teams become more efficient and reducing duplicated efforts. [15]
- **Enhanced Developer and Consumer Experience:** Contributes to a more sophisticated, streamlined, and automated experience for both the infrastructure product teams and the DevOps teams consuming the infrastructure products. [22]

Operating Model and Considerations:

The Automation and Developer Enablement Team should be structured as an **enabling platform team**. [31] Its mission is to build and maintain the developer enablement platform and a suite of shared automation tools, frameworks, and AI Ops services, acting as internal consultants and subject matter experts. It's crucial that this team empowers the product-aligned teams rather than taking over their core product automation responsibilities, which could lead to bottlenecks or dilute product team ownership. [31] The focus should be on providing capabilities that enhance the product teams' ability to deliver and manage their own services effectively, and to provide advanced, platform-wide insights and automation for consumers like DevOps teams.

4. Embedding Key Capabilities in Your New Teams

Transitioning to product-oriented teams is not just about redrawing an organizational chart; it's about embedding new capabilities and ways of working within these teams. This section details how to integrate self-service automation, full lifecycle management, robust governance, and effective Tier-3 support into the fabric of the new infrastructure product teams.

Driving Self-Service Automation and Orchestration

A cornerstone of the 'IT as a Service' model is empowering users to provision and manage infrastructure resources independently through self-service mechanisms, underpinned by robust automation and orchestration. [58]

Tools and Practices for User-Driven Infrastructure Provisioning

- **Internal Developer Portal (IDP):** This will be the primary gateway for users to access and consume infrastructure products.[23] The IDP should provide a user-friendly catalog of available services (as defined in Section 3.1.2), clear descriptions, configuration options, and a simple interface for requesting and managing resources.[38]
- **Infrastructure as Code (IaC):** All infrastructure products will be defined and managed using IaC principles. Tools like Terraform or OpenTofu will be used for provisioning diverse resources (VMs, networks, storage, cloud services), while configuration management tools like Ansible will ensure consistent setup and state.[23] This codebase becomes the single source of truth for infrastructure configurations.
- **Standardized Templates and Blueprints:** Product teams will develop and maintain a library of version-controlled, reusable IaC templates and blueprints (e.g., Terraform modules, Ansible roles and playbooks) representing common infrastructure patterns and product configurations.[23] Users will select these from the IDP, customize parameters within defined limits, and trigger automated deployment.
- **Orchestration Platforms:** To manage the complexity of end-to-end service delivery, especially in a hybrid environment, an orchestration platform will be crucial. Such platforms (e.g., Itential, Argo Workflows, or potentially custom-built solutions leveraging scripting and APIs) coordinate tasks across multiple automation tools and technology domains, enabling the full lifecycle orchestration of complex services from user request to operational readiness.[27]

The success of self-service hinges critically on the **Developer Experience (DX)**. It's

not enough to merely provide a portal with automation scripts. The entire self-service platform—its interfaces, documentation, the speed and reliability of provisioning, and the clarity of feedback—must be designed with a relentless focus on the user (primarily developers). A poor DX will inevitably lead to low adoption, users reverting to old manual processes or creating "shadow IT" solutions, thereby negating the benefits of the new model.[15] Platform engineering explicitly emphasizes enhancing DX as a key outcome.[22] This means infrastructure product teams must adopt UX/UI design principles, treat their internal developers as valued customers, and establish continuous feedback loops to iterate and improve their self-service offerings.[15] The self-service platform itself is a product that requires ongoing attention and refinement. [22]

Implementing Full Lifecycle Management for Infrastructure Products

Adopting a product mindset means taking responsibility for the entire lifecycle of each infrastructure product, from initial conception through to eventual retirement. This is a significant shift from a project-based approach where focus often wanes after initial deployment. [14]

From Planning and Development to Maintenance and Retirement

Each infrastructure product will have a defined lifecycle, actively managed by its dedicated product team [14]:

- **Planning:** Continuous engagement with internal users to understand their evolving needs, identify new product opportunities, define and update product roadmaps, forecast demand, and plan for capacity and technology evolution (e.g., new versions, alternative solutions).[14]
- **Acquisition/Development:** Product teams will build (using IaC, automation scripts, and potentially custom software components) or procure (e.g., specific hardware, cloud provider services, third-party software) the necessary components for their products. The focus will be on standardization, security by design, reusability, and cost-effectiveness.[42]
- **Deployment & Integration:** Deployment processes will be highly automated, typically initiated via the IDP or through GitOps workflows, ensuring consistent, reliable, and auditable instantiation of products into various environments (development, staging, production).[14]
- **Operation & Maintenance:** This encompasses proactive monitoring of product health, performance, and utilization; automated patching and upgrades; performance tuning; capacity management; and efficient incident response for the owned products.[14]

- **Optimization:** Product teams will continuously analyze operational data, cost metrics, security assessments, and user feedback to identify opportunities for improvement. This includes enhancing features, improving reliability, optimizing costs, and streamlining operational processes. [14]
- **Retirement/Disposal:** A formal, documented process for securely decommissioning outdated or superseded products or components is essential. This includes data migration strategies, resource cleanup (especially in cloud environments to avoid orphaned resources and costs), and communicating the retirement plan to users.[14]

This holistic responsibility requires product teams to possess or develop a blend of skills spanning software engineering (for IaC and automation), systems engineering, operations, security, and elements of product management (roadmapping, user engagement, value articulation).

Effectively managing infrastructure as enduring products with ongoing lifecycles necessitates a fundamental shift from traditional project-based funding to a **continuous, value-stream-aligned funding model**. Project-based funding, with its defined start and end dates, is ill-suited for the persistent nature of product ownership and often leads to the accumulation of technical debt as funding diminishes post-launch, leaving insufficient resources for ongoing maintenance, improvement, or adaptation.[4] The organizational restructure must therefore be accompanied by a corresponding budgetary restructure. Funding should be allocated to the product teams or the value streams they support, empowering them to manage the full lifecycle of their infrastructure products proactively and make informed trade-offs between new development, operational stability, and technical debt reduction. This represents a significant operational and financial planning evolution for a Fortune 100 company.[11]

Integrating Governance and Security by Design

In a self-service, product-oriented model, governance and security cannot be afterthoughts or manual checkpoints; they must be integral, automated aspects of the product delivery lifecycle. [58]

Policy as Code, Automated Compliance, and Risk Management

- **Codified Policies:** Security, compliance, and operational policies will be translated into machine-readable code (Policy as Code - PaC) and embedded directly into IaC templates, CI/CD pipelines, and automation workflows.[14] Tools like Open Policy Agent (OPA) can be used with Terraform, or policy features within

platforms like Spacelift or native cloud provider services (e.g., AWS Config Rules, Azure Policy) can be leveraged. This ensures that governance is applied consistently and automatically every time an infrastructure product is provisioned or modified.[14]

- **Automated Compliance:** Implement automated tools and processes to continuously check configurations against defined internal and external standards (e.g., CIS Benchmarks, NIST frameworks, SOC2, ISO 27001, or specific Fortune 100 requirements). Evidence for compliance should be gathered automatically, streamlining audit processes.[14]
- **Shift-Left Security:** Integrate automated security scanning tools directly into CI/CD pipelines for infrastructure. This includes vulnerability scanning of OS images and software components, static analysis of IaC for security misconfigurations (e.g., using tfsec, checkov), and secrets detection to prevent accidental exposure of credentials.[14]
- **Proactive Risk Management:** Gartner consistently emphasizes that robust risk management and clear governance are fundamental principles for creating adaptable and resilient IT infrastructure strategies.[33] The goal is to move towards a state of continuous compliance and a proactive, preventative security posture, rather than relying on periodic, often disruptive, manual audits.[14]

Rethinking Tier-3 Support in a Product Model

The requirement to maintain Tier-3 support within an engineering-focused organization necessitates a new approach that aligns with product ownership and operational excellence. [31]

Embedding Expertise within Product Teams

The engineers who design, build, and maintain an infrastructure product are, by definition, the foremost experts on its intricacies. Therefore, they are best positioned to provide deep, authoritative Tier-3 support when complex issues arise that cannot be resolved by lower tiers or self-service resources.[31] This means Tier-3 support becomes an integral, non-negotiable responsibility of each infrastructure product team for the specific products they own. This model directly supports the "engineering-focused" directive, as teams own their services end-to-end, including their ultimate reliability and supportability.

Practical models for managing this embedded Tier-3 responsibility include:

- **SRE-Style On-Call Rotations:** Team members within a product team rotate on-call duties specifically for their owned products. They are responsible for handling escalated incidents, performing deep troubleshooting, and

implementing fixes.[22] This model ensures that all team members develop and maintain critical operational expertise and share the support burden.
- **"You Build It, You Run It" (YBIYRI):** This philosophy, central to DevOps and SRE, reinforces that the team responsible for developing a service is also responsible for its operation and support in production.[27] This creates strong accountability and rapid learning loops.
- **Careful Use of Specialists:** While the entire team shares ownership, certain individuals might naturally cultivate deeper expertise in specific, highly complex aspects of a product. They can serve as internal escalation points for the most challenging issues.[63] However, this must be managed carefully to avoid creating single points of failure (SPOFs) or mini-silos within the team. Knowledge sharing and cross-training remain paramount.

Balancing Deep Support with Proactive Product Development

A critical challenge in embedding Tier-3 support within product teams is ensuring that reactive support work does not consume all of the team's capacity, thereby stifling proactive product development, innovation, and strategic improvement efforts. This balance is essential for the long-term health and evolution of the infrastructure products.

Strategies to maintain this crucial balance include:

- **Aggressive Automation of Operations:** A relentless focus on automating repetitive operational tasks, common incident remediation procedures, and diagnostic data gathering. This reduces manual toil and the frequency of easily preventable recurring incidents.[6]
- **Comprehensive Observability:** Implementing robust and intelligent monitoring, logging, and alerting systems. This allows for proactive issue detection, faster root cause analysis, and often, resolution before users are significantly impacted.[22]
- **Well-Defined SLOs and Error Budgets:** Establishing clear Service Level Objectives (SLOs) for each infrastructure product and utilizing error budgets. Error budgets provide a data-driven framework for teams to decide when to prioritize reliability work (if SLOs are at risk) versus new feature development (if SLOs are being comfortably met).[30]
- **Effective Lower-Tier Support & Rich Self-Service:** Ensuring that Tier-0 support (comprehensive, easily searchable self-service knowledge bases, FAQs, troubleshooting guides integrated into the IDP) and, if applicable, any existing Tier-1/2 general enterprise helpdesk functions are highly effective at resolving simpler, common issues before they escalate to the specialized product teams.[23]

- **Rigorous Blameless Post-Incident Reviews (PIRs):** Conducting thorough, blameless PIRs after significant incidents to understand the complete sequence of events, identify all contributing factors (technical, process, human), and implement robust preventative measures. The learnings and action items from PIRs must feed directly back into the product development backlog as bug fixes, architectural improvements, or new features designed to enhance resilience.[4]
- **Platform Focus on Developer Empowerment:** Platform engineering teams should prioritize creating tools, abstractions, and self-healing mechanisms that empower developers and systems to resolve many issues independently, thereby reducing the direct support burden on the infrastructure product teams.[22]

When the engineers who build a product are also responsible for its deepest level of support, a powerful, intrinsic feedback loop is created. Direct exposure to the product's failures, limitations, and user pain points provides invaluable, unfiltered data that might be diluted or lost if Tier-3 were a separate, isolated team.[27] This firsthand experience becomes a potent driver for prioritizing fixes, architectural enhancements, and ultimately, building more resilient, reliable, and user-friendly infrastructure products. This "pain-driven development" can lead to more robust outcomes faster than if development and deep support were disconnected, provided there is careful capacity management to balance reactive and proactive work.[19]

5. Navigating the Transformation: People and Processes

A successful transformation to a product-oriented infrastructure model hinges as much on people and processes as it does on technology and structure. This section addresses the critical aspects of fostering a collaborative culture, mapping and developing skills, and managing the significant organizational change involved, including the integration of all staff.

Fostering a Collaborative Culture and Effective Knowledge Sharing

The current state of "poor collaboration" and "knowledge gaps" must be actively addressed by cultivating an environment where teamwork and shared learning are paramount. [71]

Strategies for Cross-Functional Teamwork and Breaking Silos

- **Promote Open Communication, Trust, and Mutual Respect:** These are the foundational elements of any high-performing team. Leadership must actively model and encourage open dialogue, where team members feel safe to share ideas, voice concerns, and admit mistakes without fear of blame.[72] Transparency in decision-making and information sharing builds trust.[72]

- **Establish Common Goals and Shared KPIs:** Align teams around overarching objectives that require collective effort. For instance, instead of a storage team being measured solely on storage uptime and a virtualization team on VM availability, a "Compute Platform Product Team" might be measured on the end-to-end time to provision a fully functional application environment and the satisfaction of the developers using that platform.[72] This encourages a shared sense of responsibility for the overall service.
- **Implement Structured Collaboration Forums:** Regular cross-team meetings, product demos (where teams showcase their work to each other and to users), joint planning sessions, and shared retrospectives are essential for maintaining alignment, sharing progress and learnings, identifying and managing inter-product dependencies, and collaboratively resolving issues.[71]
- **Leverage Collaboration Tools Effectively:** Utilize shared knowledge bases (e.g., Confluence, SharePoint), integrated communication platforms (e.g., Slack, Microsoft Teams), and product/project management tools (e.g., Jira, Azure DevOps) that provide visibility across teams and facilitate seamless information exchange.[71]
- **Establish Communities of Practice (CoPs) or Guilds:** These are cross-functional groups of individuals who share a common interest or expertise, regardless of which product team they belong to. For example, a "Kubernetes CoP," an "Automation Guild," or a "Security Champions Guild" can facilitate organic knowledge sharing, the development of best practices, peer support, and the consistent application of standards across different product teams.[18] This helps maintain technical coherence and continuous learning across the organization.

While structural changes can be designed and implemented, shifting the deeply ingrained culture from siloed expertise and a reactive operational focus to one of collaborative product ownership and proactive engineering excellence will be the most challenging and time-consuming aspect of this transformation. This requires sustained, visible leadership effort, patience, and consistent reinforcement of desired behaviors. Overcoming resistance to change, addressing fears of losing specialist status, and helping individuals learn new collaborative ways of working demand more than just new organizational charts; they require a fundamental shift in mindset, actively championed from the top.[4]

Mapping Skills and Cultivating Talent for New Roles

The transition to product-oriented teams necessitates a significant evolution in the

skills and competencies of the existing 24-person infrastructure team. [27]

Identifying Skill Gaps and Transition Paths

A crucial early step is to conduct a comprehensive **skills inventory** of the current team. This involves mapping existing technical expertise, certifications, and soft skills against the anticipated requirements of the new product-oriented roles (e.g., Platform Engineer, Infrastructure Product Engineer, Site Reliability Engineer, Cloud Engineer, Product Owner for Infrastructure Products).[26]

Traditional infrastructure roles will transform significantly:

- A **Storage Administrator**, for example, might transition into a **"Storage Product Engineer"** role within an "Enterprise Storage Solutions" product team. This new role would focus on delivering various storage services (block, file, object, backup) as self-service, API-driven products. Beyond deep expertise in specific storage technologies, this requires proficiency in automation scripting (e.g., Python, PowerShell, Go), Infrastructure as Code tools (e.g., Terraform, Ansible), API design and consumption, data protection strategies, and familiarity with cloud storage services (e.g., AWS S3, Azure Blob Storage, Google Cloud Storage).[27]
- **OS Administrators (Windows, Linux)** could evolve into **"Compute Platform Engineers"** or become specialized OS experts within product teams that consume these OS services (e.g., an application platform team). They will need to significantly enhance their skills in advanced automation, declarative configuration management (e.g., Ansible, Puppet, PowerShell DSC), managing OS instances in public cloud environments (AWS EC2, Azure VMs), containerization technologies (Docker, Kubernetes), and potentially immutable infrastructure concepts and security hardening at scale. [27]
- **Virtualization Engineers** (e.g., VMware specialists) will likely become key members of **"Private Cloud Platform Teams"** or contribute to hybrid **"Compute Platform Teams."** Their focus will shift from manual VM administration to automating the virtualization fabric, managing its APIs for consumption by higher-level orchestration tools, and ensuring its integration with cloud management platforms and the overall IDP. [26]
- Existing **Public Cloud Specialists** will be invaluable. They will likely be embedded within various product teams that have a hybrid remit (e.g., a "Compute Platform Team" managing both on-prem VMs and cloud instances, or a "Cloud Gateway Services Team"), bringing deep expertise in specific cloud provider services (AWS, Azure, GCP) and helping to design and implement solutions that bridge the on-premises and public cloud environments. [27]

Beyond purely technical skills, the most critical shift for many traditional infrastructure engineers will be the development of **"product thinking."** This involves understanding internal user needs, defining value propositions for their infrastructure services, thinking in terms of product lifecycles, using data to drive decisions, and effectively communicating the benefits and usage of their products.[71] This is a significant mindset evolution from managing "boxes and software" to owning and evangelizing a service or product consumed by others.

Proposed Table: Illustrative Skills Transition Matrix

The following table provides a structured way to visualize potential transitions and skill development needs [27]:

Traditional Role	Potential New Product Team Role	Current Key Skills (Illustrative)	New Skills Required for Product Model (Illustrative)	Suggested Development/Training Focus	Potential Challenges in Transition
Senior Linux Administrator	Compute Platform Engineer (Linux Focus)	Deep Linux OS, Bash scripting, LVM, Patching	Python/Go, Ansible/Terraform, Kubernetes/Docker, AWS/Azure Linux instances, Monitoring tools (Prometheus), Git, CI/CD concepts, Agile methodologies	Cloud provider certifications (AWS/Azure Linux), CKA/CKAD, Advanced Ansible/Terraform courses, Internal workshops on product mindset & Agile	Shift from imperative to declarative automation, Adapting to ephemeral/immutable concepts, Broader system view beyond OS
Storage Architect/Admin	Storage Product Engineer	SAN/NAS expertise (e.g., NetApp, EMC), FC, iSCSI, Backup	Cloud storage (S3, Azure Blob), IaC for storage (Terraform), Automation (Python/Ansi	Vendor-specific cloud storage training, Terraform/Ansible for storage automation,	Moving from hardware-centric to service/API-centric view, Learning software development

		software	ble for storage APIs), API design basics, Data protection strategies for cloud, Product ownership basics	Workshops on defining service tiers and SLAs, Introduction to product management	practices for IaC, Understanding diverse application storage needs
Network Engineer	Network Automation Engineer / Network Product Engineer	Cisco/Juniper routing & switching, Firewalls, Load balancers, BGP, OSPF	Python/Ansible for network automation, Cloud networking (VPCs, VNETs, Security Groups, Cloud Load Balancers), IaC for network (Terraform), API integration, NetDevOps principles	CCNA/CCNP Cloud or Data Center, Python for network engineers (e.g., Nornir, Netmiko), Training on cloud provider networking services, Terraform for networking	Transitioning from CLI-driven management to API/automation-driven, Understanding software-defined networking (SDN) concepts, Security implications of automated network changes
Virtualization Engineer (VMware)	Private Cloud Platform Engineer / Hybrid Compute Engineer	vSphere suite (ESXi, vCenter, NSX-T), PowerCLI, Storage integration	vRealize Automation/ Aria, Terraform Provider for vSphere, Kubernetes on vSphere (Tanzu), Public cloud IaaS (AWS EC2, Azure VMs), Hybrid	VMware VCP-CMA, VCP-DCV, Cloud provider certifications, Terraform/Ansible, Workshops on building self-service catalogs for	Abstracting vSphere into consumable services, Integrating with broader cloud orchestration, Shifting focus from infrastructure components

			cloud management tools (e.g., Azure Arc), Python	VMs	to platform capabilities
Public Cloud Engineer (e.g., AWS)	Cloud Platform Engineer / SRE (Cloud Focus)	AWS EC2, S3, VPC, IAM, CloudFormation/CDK, Lambda	Multi-cloud proficiency (Azure/GCP if needed), Advanced Kubernetes (EKS, AKS, GKE), Advanced IaC (Terraform), FinOps principles, Advanced security in cloud, SRE practices (SLOs, error budgets)	Advanced cloud certifications (AWS Solutions Architect Professional, Azure Expert), Kubernetes certifications (CKA/CKS), SRE training, FinOps certification	Deepening expertise beyond one cloud, Applying SRE principles systematically, Leading on cost optimization and governance in a product context
Windows OS Administrator	Compute Platform Engineer (Windows Focus)	Windows Server, Active Directory, PowerShell, GPO, SCCM	Azure AD, PowerShell DSC/Ansible for Windows, Windows containers, Azure IaaS (Windows VMs), Security hardening for cloud, Git, CI/CD for Windows configurations	Microsoft Azure certifications (AZ-104, AZ-800/801), Advanced PowerShell/DSC courses, Training on Ansible for Windows, Workshops on IaC for Windows environments	Adapting to cloud-based identity (Azure AD), Managing Windows in IaC/DevOps paradigms, Containerizing Windows applications
Backup	Data	Backup	Cloud-native	Certification	Shifting from

Administrator	Protection Product Engineer	software (e.g., Commvault, Veeam), Tape libraries, DRP	backup solutions (AWS Backup, Azure Backup), BaaS/DRaaS concepts, IaC for backup policies, Automation of recovery testing, Ransomware protection strategies	s for cloud backup solutions, Training on scripting for backup automation, Workshops on modern DRP and cyber resiliency	on-prem backup tools to cloud-integrated services, Automating DR validation, Understanding application-consistent backup in distributed environments
Automation / Developer Enablement Focus	Automation and Developer Enablement Engineer	Scripting (Python, Bash), CI/CD Tooling, Basic IaC, Observability Concepts	Advanced Automation Frameworks, AI Ops Tooling, Platform Development (IDP), Advanced IaC (Terraform, Ansible), Kubernetes, API Design, Developer Advocacy, Product Mindset for Internal Tools	Advanced IaC/Automation Certifications, Kubernetes (CKA/CKAD), AI Ops Training, Platform Engineering Workshops, UX for Developer Tools	Balancing enablement with direct platform development, Scaling automation efforts across diverse product teams, Keeping pace with AI Ops evolution
General IT Support/Unmanaged	Junior Platform Engineer / Product Support Specialist (within	Varied, potentially basic troubleshooting, hardware	Foundational Linux/Windows, Basic scripting (Bash/PowerShell), Ticketing	CompTIA A+/Network+/Security+, Entry-level cloud certifications (AWS CCP,	Defining a clear role and growth path, Acquiring foundational technical

	Product Team)	setup	systems, Customer service skills, Understanding of specific product domain, Monitoring tool usage, Documentation writing	Azure Fundamentals), Mentorship by senior team members, Structured training on core platform tools and products, ITIL Foundation	skills for automation and cloud, Integrating into a structured engineering team, Overcoming potential disengagement

Value of this Matrix: This matrix is a critical tool for the IT Executive. It directly addresses the "knowledge gaps" concern and provides a structured framework for managing the human capital aspect of this significant transformation. It helps visualize potential career paths for existing staff, identify specific retraining needs, and inform recruitment strategies if external talent is required. It makes the people-side of the change tangible, actionable, and less daunting.[26]

Cultivating Talent

A proactive talent development strategy is essential for the success of this reorganization:

- **Invest in Targeted Training:** Provide access to and funding for relevant training programs, industry certifications (e.g., cloud provider certifications, Kubernetes CKA/CKAD, Terraform Associate, SRE Foundation), and hands-on workshops to upskill and reskill existing staff in areas like automation, cloud technologies, IaC, and SRE principles.[4]
- **Encourage Cross-Training and Mentorship:** Facilitate internal cross-training initiatives, pair programming, and mentorship programs where experienced team members can guide others. Temporary job rotations within and between the newly formed product teams can also broaden skillsets, improve understanding of different product domains, and break down residual knowledge silos.[72]
- **Foster a Culture of Continuous Learning:** Promote an environment where continuous learning, experimentation, and adaptation are core values. Provide resources (e.g., subscriptions to online learning platforms, time for self-study) and create psychological safety, where team members feel empowered to learn

new technologies and approaches, even if it involves occasional setbacks or "failing fast".[4] Microsoft's own IT transformation journey underscores the importance of embracing new roles, skills, and a growth mindset when moving to modern engineering practices.[2]

Effective Change Management and Onboarding

Managing the human side of this transformation is paramount, especially given the existing team dynamics and the inclusion of previously unmanaged members. [2]

Integrating All Staff, Including Previously Unmanaged Members

A clear, consistent, and structured onboarding plan must be developed for **all** team members transitioning into the new organizational structure and its associated ways of working. This applies regardless of their previous role, seniority, or management status.[2]

The existence of "unmanaged members" presents a specific challenge but also a significant opportunity. This reorganization is the precise moment to formally integrate these individuals into the team structure. This involves:

1. Clearly defining their roles and responsibilities within a specific product team. [8]
2. Assessing their current skills and identifying any gaps relative to their new role. [22]
3. Providing them with the necessary support, training, mentorship, and direction. [63]
4. Setting clear performance expectations and providing regular feedback. [8] This process is not just about filling slots in an org chart; it's about ensuring every individual feels valued, understands their contribution, and is equipped to succeed in the new model. Leaving staff unmanaged is a risk to productivity, morale, and governance; this transformation provides the impetus to address this directly, potentially unlocking hidden talents or, conversely, revealing performance issues that need to be managed constructively.[22]

Communicating the Vision and Managing Resistance

Change of this magnitude will inevitably encounter resistance. Proactive and empathetic change management is key:

- **Clearly Articulate the "Why":** Leadership must consistently and clearly communicate the vision behind the reorganization – the compelling reasons for the change and the anticipated benefits for the individuals, the team, the IT department, and the broader Fortune 100 company.[4] Focus on the opportunities for skill growth, increased impact, and more engaging work.
- **Involve Teams Appropriately:** Where feasible, involve team members in the

design and planning of aspects of the change that directly affect them. This fosters a sense of ownership and can significantly reduce resistance.[22] For instance, teams could contribute to defining the working agreements or initial backlogs for their new product areas.

- **Address Resistance Proactively and Empathetically:** Acknowledge that change can be unsettling. Create safe channels for team members to voice concerns, ask questions, and provide feedback. Address these concerns transparently and provide support to help individuals navigate the transition. Highlight success stories and quick wins from early pilots to build confidence and momentum.[22]
- **Visible Leadership Commitment:** Unwavering and visible commitment from IT leadership is paramount. Leaders must champion the new model, consistently reinforce the desired behaviors and cultural shifts, allocate necessary resources for training and tools, and make tough decisions when conflicts arise between old and new ways of working.[4]

6. Measuring Success and Driving Continual Improvement

The transition to a product-oriented infrastructure model is not a one-time project but an ongoing journey of evolution and refinement. Establishing relevant metrics and fostering a culture of continual improvement are critical for ensuring the long-term success and adaptability of the new organization.

Establishing Key Performance Indicators (KPIs) for Infrastructure Product Teams and Services

The success of the new model requires a shift in how performance is measured. Traditional operational metrics (e.g., server uptime, number of tickets closed) while still having a place, must be augmented, and in some cases superseded, by product-oriented and value-driven KPIs that reflect the new strategic priorities.[4] The choice of KPIs will powerfully signal what is truly valued in the new organization. Focusing solely on technical output metrics (e.g., number of VMs deployed per month) without measuring user satisfaction, adoption rates, or the impact on developer velocity will undermine the fundamental shift to a product mindset. A balanced scorecard approach is therefore essential.

Examples of relevant KPIs include:

- **Product Adoption & Usage:**
  - *Adoption Rate:* Percentage of target internal users (e.g., development teams) actively consuming specific infrastructure products. [17]

    - *Consumption Volume:* Growth in the utilization of self-service products (e.g., number of API calls to provisioning services, resources deployed via IDP).
- **Self-Service & Automation Efficiency:**
    - *Self-Service Success Rate:* Percentage of service requests successfully fulfilled via automated self-service channels without requiring manual intervention from the product team.
    - *Time to Provision:* Average cycle time for users to obtain requested infrastructure resources through self-service (e.g., time from request to VM readiness).
    - *Reduction in Manual Toil:* Quantifiable decrease in time spent by product teams on repetitive operational tasks. [15]
- **User Satisfaction & Experience:**
    - *Developer/User Satisfaction Scores:* Measured via surveys (e.g., Net Promoter Score - NPS, Customer Satisfaction - CSAT) regarding the quality, usability, reliability, and support for infrastructure products. [58]
    - *Ease of Use Metrics:* Feedback on the intuitiveness and efficiency of the IDP and self-service interfaces. [17]
- **Service Reliability & Quality:**
    - *Service Level Objective (SLO) Adherence:* Tracking performance against defined SLOs (availability, latency, error rates) for each infrastructure product. [28]
    - *Change Failure Rate:* Percentage of changes to infrastructure products that result in incidents or require remediation. [14]
    - *Mean Time To Recovery (MTTR):* Average time taken to restore service after an incident affecting an infrastructure product.
- **Cost Efficiency & Value:**
    - *Cost per Product Unit:* Tracking the operational cost of delivering each infrastructure product (e.g., cost per VM-hour, cost per TB of managed storage).
    - *Resource Optimization:* Evidence of efficient use of underlying resources (e.g., consolidation ratios, cloud spending optimization). [58]
- **Impact on Business Agility (Potentially more qualitative initially):**
    - *Feedback from Business Units/Application Teams:* Testimonials or survey data on how the new infrastructure services are enabling faster project delivery, quicker iteration cycles, or reduced lead times for new features.

These KPIs should be regularly reviewed, transparently reported, and used to drive data-informed decisions within the product teams and by IT leadership. They must directly align with the overarching goals of 'IT as a Service,' self-service automation,

fostering a product mindset, and ensuring robust governance.

Adopting an Iterative Approach to Refine the Operating Model

The initial reorganization and implementation of product teams should be viewed as Version 1.0 of the new operating model. It is crucial to embrace a mindset of continual improvement, recognizing that the structures, processes, and product offerings will need to adapt and evolve based on experience, feedback, and changing business needs.[39] The ITIL 4 framework offers valuable guiding principles for this ongoing refinement [39]:

- **Focus on Value:** Continuously assess whether the infrastructure products and team activities are delivering tangible value to internal customers and the business. [39]
- **Start Where You Are:** Acknowledge the current state and build upon existing strengths and processes where appropriate, rather than attempting a complete "rip and replace" of everything. [39]
- **Progress Iteratively with Feedback:** Implement changes in manageable increments. Establish robust feedback loops with users, product teams, and other stakeholders to gather insights and make data-driven adjustments to the operating model, product features, and team structures. [39]
- **Collaborate and Promote Visibility:** Maintain open communication and transparency about the performance of the new model, ongoing improvement initiatives, and any planned changes. [39]
- **Think and Work Holistically:** Consider the entire infrastructure service value system and how different product teams and capabilities interconnect to deliver end-to-end value. [39]
- **Keep It Simple and Practical:** Avoid unnecessary complexity in processes and tools. Focus on solutions that are straightforward, effective, and aligned with business goals. [39]
- **Optimize and Automate:** Continuously seek opportunities to optimize processes, improve efficiency, and further automate tasks across the infrastructure product lifecycle. [39]

For teams to genuinely embrace "progressing iteratively with feedback" and to effectively "optimize and automate," a culture of **psychological safety** and a **blameless approach to failures** is indispensable. Team members need to feel secure in experimenting with new approaches, identifying inefficiencies or failures in existing products or processes, and proposing changes without fear of personal blame or retribution.[4] If the prevailing culture is punitive or overly resistant to change, valuable feedback will be suppressed, learning will be stifled, and genuine improvement efforts

will stall. Leadership plays a critical role in actively cultivating an environment where transparency about problems is encouraged, and teams are empowered to learn from setbacks and drive improvements in their products and processes. This is particularly important when moving away from established "siloed expert" domains, where individuals might feel their traditional expertise or status is being challenged by new models.

Regular reviews of team performance against KPIs, periodic reassessments of product roadmaps in light of user feedback and business strategy, and formal retrospectives on the overall effectiveness of the operating model should be institutionalized practices.

7. Strategic Recommendations and Phased Implementation Roadmap

Successfully transforming a traditional, siloed infrastructure organization into an agile, product-oriented function requires a deliberate, phased approach, underpinned by clear strategic choices and strong leadership.

Key Strategic Choices and Actionable Recommendations

1. **Prioritize the Definition of 2-3 Pilot Infrastructure Products:** Instead of attempting a "big bang" transformation across all services simultaneously, begin by selecting 2-3 infrastructure services that have high potential impact, visibility, and are good candidates for productization. Use these as pilots to test and refine the new team structures, processes, product management practices, and self-service capabilities.[71] These pilot products should ideally address significant existing pain points for internal users, making their value proposition clear and compelling.
2. **Establish a Cross-Functional "Transformation Team" or "Platform Center of Excellence (CoE)":** Form a dedicated, empowered team to champion and guide the overall transformation. This team should comprise influential members from different existing silos, individuals with a strong product mindset, and potentially an experienced product manager or owner. Their role will be to define initial standards for product management, platform engineering, governance, and tooling, as well as to coach and support the first pilot product teams.[17] This central group can ensure consistency and drive the adoption of new practices.
3. **Invest Heavily in Skills Development and Change Management from Day One:** Recognize that this is primarily a people and culture transformation, enabled by technology and structure. Allocate significant resources and

leadership attention to comprehensive skills development programs (technical skills, product mindset, agile methodologies, collaboration) and a robust change management strategy from the very beginning of the initiative. This is not an afterthought but a critical prerequisite for success. [2]

4. **Select and Standardize Key Automation, Orchestration, and IDP Tooling:** Based on the defined product strategy and architectural principles for hybrid cloud, make strategic decisions on the core set of tools that will underpin the Internal Developer Platform. This includes IaC tools (e.g., Terraform), configuration management (e.g., Ansible), orchestration engines, monitoring and observability platforms, and the IDP interface itself. Provide comprehensive training and support for these standardized tools to ensure consistency and proficiency. [14]
5. **Develop and Execute a Clear, Continuous Communication Plan:** Keep all stakeholders, especially the 24 team members directly impacted by the reorganization, consistently informed and engaged throughout the process. Communicate the vision, the rationale, the progress, the successes, and the challenges openly and frequently. Provide multiple channels for feedback and questions. [2]

A High-Level Roadmap for Phasing the Reorganization

The following phased roadmap provides a structured approach to implementation:

Phase 1: Foundation & Pilot (Duration: 0-6 Months)

- **Activities:**
  - Conduct a detailed assessment of the current state: skills inventory of the 24 team members, existing processes, tools, and pain points. [26]
  - Define the initial internal infrastructure product portfolio and service catalog (as per Section 3.1). Select 2-3 high-impact services for the pilot phase. [71]
  - Form the first 2-3 pilot product teams (max 8 people each), carefully selecting team members who are adaptable and enthusiastic. Clearly define their product ownership, roles, and initial backlog. [71]
  - Establish the Transformation Team/Platform CoE. [17]
  - Develop and deliver initial training plans for pilot teams focusing on product mindset, agile practices, core automation tools, and collaboration. [22]
  - Define the initial governance framework, including principles for policy-as-code and security integration for pilot products. [13]
  - Begin development of basic self-service capabilities (e.g., simple portal interface, initial IaC templates) for the pilot products. [14]
  - Establish baseline KPIs for pilot products. [4]

- **Key Outcomes:** Pilot product teams operational, initial product definitions and roadmaps for pilots, foundational IDP components in development, early learnings captured.

Phase 2: Expansion & Refinement (Duration: 6-12 Months)

- **Activities:**
  - Formally launch the pilot infrastructure products to a controlled group of internal users.
  - Actively gather user feedback and performance data from the pilots. [17]
  - Conduct retrospectives with pilot teams and the Transformation Team to iterate on team structures, processes, product offerings, and IDP capabilities based on learnings. [6]
  - Begin expanding the product-oriented model to additional infrastructure service areas, forming new product teams.
  - Roll out broader skills development programs across the entire infrastructure organization. [22]
  - Mature the self-service portal and expand the library of automated product offerings. [14]
  - Fully integrate Tier-3 support responsibilities within all operational product teams, including on-call rotations and incident management processes. [23]
  - Refine and expand the governance framework (e.g., more automated policy checks). [16]
- **Key Outcomes:** Successful delivery and iteration of pilot products, validated operating model for product teams, expanded product portfolio under new model, enhanced IDP capabilities, measurable improvements in pilot KPIs.

Phase 3: Scale & Optimize (Duration: 12-18+ Months)

- **Activities:**
  - Transition all remaining infrastructure services to the product-oriented model, ensuring all services are managed as products by dedicated, appropriately sized product teams.
  - Achieve comprehensive self-service capabilities with deeply embedded governance for the majority of infrastructure offerings. [14]
  - Ensure mature lifecycle management processes are in place and actively practiced by all product teams. [14]
  - Focus on continuous improvement: product teams regularly optimize their products and processes based on KPIs, user feedback, and evolving business needs. [39]
  - Deepen and sustain a strong product and engineering culture across the

entire infrastructure organization. [4]
    - Explore advanced platform capabilities (e.g., AIOps, more sophisticated orchestration). [27]
- **Key Outcomes:** Entire infrastructure organization operating under the new product-centric model, high adoption of self-service capabilities, demonstrable improvements in agility, efficiency, and user satisfaction, a culture of continuous improvement and innovation.

This level of transformation, impacting organizational structure, individual roles, essential skills, operational processes, cultural norms, and potentially funding models, is a significant undertaking. Therefore, **unwavering, visible, and sustained executive sponsorship** is not merely beneficial but absolutely non-negotiable for success, especially within the complex environment of a Fortune 100 company.[4] This sponsorship is essential to champion the new vision, secure the necessary resources for training and technology, consistently reinforce new behaviors and mindsets, navigate organizational inertia, and make decisive interventions when conflicts arise between the old and new ways of working. The IT Executive initiating this change must be prepared to actively lead and drive this commitment throughout the entire transformation journey.

8. Conceptual Organizational Diagrams

To provide further context and aid in visualizing the proposed changes, the following conceptual diagrams illustrate the new organizational model and the mapping of existing skills to this new structure. These are high-level representations and would be further detailed during the transformation planning phases.

8.1 Diagram: Conceptual New Organizational Model

This diagram illustrates a potential structure for the reorganized hybrid infrastructure engineering organization, emphasizing product-aligned teams and the integrated "Automation and Developer Enablement Team."

```
                    IT Leadership
|
            ---------------------------------
| |
```

Automation and Developer Enablement Team Product-Aligned Infrastructure Teams
(Central Enabling Function, <8 people) (Owning services end-to-end, <8 people each)
| |
- Owns Automation & Developer Enablement |--- Compute Platform Services Product Teams
Platform Products (CI/CD, IaC Catalog, (e.g., Linux VM Service Team, Windows VM Service Team,
Observability Services, Artifact Repositories) Container Platform Team)
- Automation CoE & Frameworks |
- AI Ops Research, Tooling & Enablement |--- Enterprise Storage Solutions Product Teams
- Cross-Product Automation Strategy (e.g., Block Storage Team, File Storage Team,
- Developer Experience & Enablement Object Storage Team, Data Protection Team)
| |
|--- Network Services & Automation Product Teams
(e.g., Virtual Networking Team, Load Balancing Team,
DNS/IPAM Team)
|
|--- Identity & Access Management (IAM) Product Teams
(e.g., Directory Services Team, PAM Team)
|
|--- Cloud Gateway Services Product Teams
(e.g., Cloud Account Governance Team,
Cloud Cost Management Team)

Key Principles:
- Each Product Team (including Automation and Developer Enablement) is cross-functional and responsible for the full lifecycle of its product(s).
- All teams are sized at 8 people or less.
- The Automation and Developer Enablement Team provides core platform services for developers, drives automation standards and AI Ops exploration, and supports other Product Teams.
- Collaboration mechanisms (CoPs, Guilds) exist across all teams.

8.2 Diagram: Illustrative Skills Mapping to New Product-Oriented Roles

This table provides a conceptual mapping of traditional IT skill areas to the new product-oriented roles and focus areas within the reorganized structure. It highlights the evolution of skills required.

Traditional Skill Area / Role Focus	Illustrative New Product Team Role / Focus Area	Key Skill Transformations Needed
OS Administration (Linux, Windows)	Compute Platform Engineer; OS Specialist within Product Teams	Advanced Automation (IaC, scripting like Python/Go, Ansible), Cloud OS Instance Management (AWS EC2, Azure VMs), Containerization (Docker, Kubernetes), Immutable Infrastructure Concepts, Security Hardening at Scale, Git, CI/CD.
Storage Administration (SAN/NAS, Backup)	Storage Product Engineer; Data Protection Product Engineer	Cloud Storage Services (S3, Azure Blob), IaC for Storage (Terraform), Storage API Automation (Python/Ansible), Service Tier Definition & SLAs, Cloud-native Backup Solutions (AWS Backup, Azure Backup), BaaS/DRaaS Concepts, Ransomware Protection.
Network Engineering (Routing, Switching, Firewalls)	Network Automation Engineer; Network Product Engineer	Cloud Networking (VPCs, VNETs, Security Groups, Cloud Load Balancers), Network Automation (Python, Ansible, Nornir, Netmiko), IaC for Network (Terraform), API Integration, NetDevOps Principles, SDN Concepts.
Virtualization Engineering (e.g., VMware)	Private Cloud Platform Engineer; Hybrid Compute Platform Engineer	Virtualization Fabric Automation (vRealize/Aria Automation), IaC for vSphere (Terraform Provider), Kubernetes on vSphere (Tanzu), Public Cloud IaaS

		(AWS EC2, Azure VMs), Hybrid Cloud Management Tools (Azure Arc), API-driven service delivery.
Public Cloud Specialization (AWS, Azure, GCP)	Cloud Platform Engineer; SRE (Cloud Focus); Embedded Cloud Specialist in Product Teams	Multi-Cloud Proficiency, Advanced Kubernetes (EKS, AKS, GKE), Advanced IaC (Terraform), FinOps Principles, Advanced Cloud Security & Governance, Systematic SRE Practices (SLOs, Error Budgets, Toil Reduction).
Active Directory Management	Identity & Access Management (IAM) Product Engineer	Azure Active Directory (Azure AD), Cloud Identity Solutions, IAM Automation (PowerShell, Graph API), Federated Identity Management, Privileged Access Management (PAM) in Hybrid Environments.
Hardware Standards & Management	(Skills integrated into relevant Product Teams, e.g., Compute, Storage)	Focus shifts from physical management to defining standards for IaC and automated provisioning, capacity planning for underlying physical infra supporting private cloud products.
Developer Tooling / Automation / AI Ops Focus	Automation and Developer Enablement Engineer (within the dedicated team)	Platform Development (IDP components like CI/CD, IaC catalogs, Observability tools), Advanced Automation Frameworks (Python, Go), AI Ops Tooling & Integration, API Design & Management, Developer Advocacy, Product Mindset for Internal Tools, Cross-team Enablement & Consulting.
General IT Support /	Junior Platform Engineer; Product Support Specialist	Foundational OS (Linux/Windows), Basic

Previously Unmanaged Roles	(within Product Team); Technical Analyst	Scripting (Bash/PowerShell), Cloud Fundamentals (AWS CCP, Azure Fundamentals), Specific Product Domain Knowledge, Customer Service Skills for Internal Users, Monitoring Tool Usage, Documentation Writing, ITIL Principles.

Note: This mapping is illustrative. Actual role transitions will depend on individual skills, aptitude for learning, and the specific needs of each product team. A significant emphasis will be placed on developing a "product mindset," automation skills, and cloud competencies across all roles.

Works cited

1. Knowledge Sharing Best Practices for Product Teams - Aha.io, accessed June 4, 2025, https://www.aha.io/roadmapping/guide/knowledge-sharing-best-practices-for-product-teams
2. Salesforce BT adopts agile customer-centric model: PwC, accessed June 4, 2025, https://www.pwc.com/us/en/library/case-studies/salesforce-business-technology.html
3. IT Support Levels Clearly Explained: L1, L2, L3 & More – BMC ..., accessed June 4, 2025, https://www.bmc.com/blogs/support-levels-level-1-level-2-level-3/
4. The Case for Adopting a Product Operating Model - Planview Blog, accessed June 4, 2025, https://blog.planview.com/the-case-for-adopting-a-product-operating-model/
5. The Product-Centric IT Operating Model Is Upon Us | Forrester, accessed June 4, 2025, https://www.forrester.com/report/the-product-centric-it-operating-model-is-upon-us/RES164936
6. Demystifying Product Centric Value Delivery with Forrester | Infosys, accessed June 4, 2025, https://www.infosys.com/services/agile-devops/insights/demystifying-product-centric-value.html
7. Continuous Improvement for Reliable Service - Google SRE, accessed June 4, 2025, https://sre.google/workbook/engagement-model/
8. From Project to Product-Centric Transformation - A New Model for IT ..., accessed June 4, 2025, http://www.valuemomentum.com/blogs/from-project-to-product-centric-transformation-a-new-model-for-it-delivery/
9. 10 IT Onboarding Best Practices for a Smoother Transition, accessed June 4,

2025, https://cloudviewpartners.com/it-onboarding-best-practices/
10. 7 best practices for team collaboration in it | Build a trusted IT culture ..., accessed June 4, 2025, https://lumenalta.com/insights/7-best-practices-for-team-collaboration-in-it
11. The IT department: structure, roles and responsibilities - Prey, accessed June 4, 2025, https://preyproject.com/blog/the-it-department
12. 10 Ways to Organize Your Product Team Structure - Product School, accessed June 4, 2025, https://productschool.com/blog/productstrategy/product-team-structure
13. Hybrid Cloud Strategies: A Practical Guide for CIOs and IT Managers, accessed June 4, 2025, https://blog.trginternational.com/hybrid-cloud-strategies-cio-it-managers
14. The Illusion of Infrastructure Automation Maturity - Spacelift, accessed June 4, 2025, https://spacelift.io/blog/the-illusion-of-infrastructure-automation-maturity
15. The Graphic Future Of IT Management - Forrester, accessed June 4, 2025, https://www.forrester.com/blogs/the-graphic-future-of-it-management/
16. EFFECTS OF IT INFRASTRUCTURE SERVICES ON BUSINESS PROCESS IMPLEMENTATION - DiVA portal, accessed June 4, 2025, https://www.diva-portal.org/smash/get/diva2:1312384/FULLTEXT01.pdf
17. Case Studies: Three customer platform engineering ..., accessed June 4, 2025, https://learn.microsoft.com/en-us/platform-engineering/case-study
18. 4 major Product organization Models - Thiga Media, accessed June 4, 2025, https://www.media.thiga.co/en/en/4-major-product-organization-models
19. Breaking Down Data Silos: Fostering Collaboration with VMware Cloud Foundation Training, accessed June 4, 2025, https://blogs.vmware.com/learning/2024/09/12/breaking-down-data-silos-fostering-collaboration-with-vmware-cloud-foundation-training/
20. The Ultimate Guide to Internal Product Management | airfocus, accessed June 4, 2025, https://airfocus.com/resources/guides/internal-pm/
21. The Importance of DevOps Team Structure | Atlassian, accessed June 4, 2025, https://www.atlassian.com/devops/frameworks/team-structure
22. How to Build a Platform Engineering Team That Scales with You, accessed June 4, 2025, https://www.opslevel.com/resources/how-to-build-a-platform-engineering-team-that-scales-with-you
23. How to Build a Platform Engineering Team - Spacelift, accessed June 4, 2025, https://spacelift.io/blog/how-to-build-a-platform-engineering-team
24. Tier 3 Explained - Turabit, accessed June 4, 2025, https://turabit.com/tier-3-explained/
25. How to Successfully Build Multi-Skilled Software Teams – DASA, accessed June 4, 2025, https://www.dasa.org/blog/how-to-successfully-build-multi-skilled-software-teams/
26. How To Implement an Effective DevOps Team Structure, accessed June 4, 2025, https://serveracademy.com/blog/understanding-and-implementing-an-effective-

devops-team-structure/
27. Inside the 2025 Gartner Guide: Why Product Thinking Is Reshaping ..., accessed June 4, 2025, https://www.itential.com/blog/company/infrastructure-orchestration/inside-the-2025-gartner-guide-why-product-thinking-is-reshaping-infrastructure-delivery/
28. Hybrid Cloud Strategy: A Guide to Business Growth - The CTO Club, accessed June 4, 2025, https://thectoclub.com/news/hybrid-cloud-strategy/
29. The Secrets to Navigating Organizational Change Smoothly, accessed June 4, 2025, https://www.cbtnuggets.com/blog/it-careers/organizational-change-it-teams
30. ITIL Framework Guide: Core Principles & Best Practices | Atlassian, accessed June 4, 2025, https://www.atlassian.com/itsm/itil
31. ITIL service catalog guide: ITSM examples and templates - ManageEngine, accessed June 4, 2025, https://www.manageengine.com/products/service-desk/itsm/what-is-service-catalog.html
32. Dismantling the Silo Mentality in Tech Companies - Techblocks, accessed June 4, 2025, https://tblocks.com/articles/dismantling-the-silo-mentality-in-tech-companies/
33. Design a Resilient IT Infrastructure Strategy - Gartner, accessed June 4, 2025, https://www.gartner.com/en/infrastructure-and-it-operations-leaders/topics/it-infrastructure
34. IT Infrastructure: Definition, Components & Optimization - Atlassian, accessed June 4, 2025, https://www.atlassian.com/itsm/it-operations-management/it-infrastructure
35. IT Infrastructure Explained: Definition, Components, and Types - Stefanini, accessed June 4, 2025, https://stefanini.com/en/insights/articles/it-infrastructure-explained-definition-components-and-types
36. What Is an Internal Developer Platform? - DuploCloud, accessed June 4, 2025, https://duplocloud.com/blog/what-is-an-internal-developer-platform/
37. Towards Model-Based Modular Product Families in the Context of Systems Engineering | The Design Society, accessed June 4, 2025, https://www.designsociety.org/download-publication/47600/Towards+Model-Based+Modular+Product+Families+in+the+Context+of+Systems+Engineering
38. What is IaaS (Infrastructure as a Service)? | Google Cloud, accessed June 4, 2025, https://cloud.google.com/learn/what-is-iaas
39. Infrastructure-as-a-Service (IaaS) - IBM, accessed June 4, 2025, https://www.ibm.com/think/topics/iaas
40. What is Infrastructure as a Service (IaaS)? - Microsoft Azure, accessed June 4, 2025, https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-iaas
41. IT infrasctructure skills matrix template - AG5, accessed June 4, 2025, https://www.ag5.com/templates/it-infrastructure-skills-matrix-template/

42. Infrastructure Lifecycle Management Best Practices - ZenGRC, accessed June 4, 2025, https://www.zengrc.com/blog/infrastructure-lifecycle-management-best-practices/
43. IT Support Reorganization: Shift Management Success Case Studies - myshyft.com, accessed June 4, 2025, https://www.myshyft.com/blog/it-support-team-reorganization/
44. Top examples of service catalogs in action - Port IO, accessed June 4, 2025, https://www.port.io/blog/top-examples-of-service-catalogs-in-action
45. Platform Engineering Team Structure: DevOps Skills + New Roles to Hire for Your Platform Engineering Team | Puppet, accessed June 4, 2025, https://www.puppet.com/blog/platform-engineering-teams
46. Breaking Down Silos: How to Foster Cross-Functional Collaboration in Software Development - Full Scale, accessed June 4, 2025, https://fullscale.io/blog/cross-functional-collaboration-development-product-design/
47. How product teams can quarterback high-performing organizations to unlock value - PwC, accessed June 4, 2025, https://www.pwc.com/us/en/services/consulting/business-transformation/library/product-operating-model-framework.html
48. Team Topologies to Structure a Platform Team | Mia-Platform, accessed June 4, 2025, https://mia-platform.eu/blog/team-topologies-to-structure-a-platform-team/
49. Cloud Operating Models: How to Build It + Common Challenges - Wiz, accessed June 4, 2025, https://www.wiz.io/academy/modern-cloud-operating-model
50. Azure hybrid options - Azure Architecture Center | Microsoft Learn, accessed June 4, 2025, https://learn.microsoft.com/en-us/azure/architecture/guide/technology-choices/hybrid-considerations
51. What is Lifecycle Management? A Complete Guide - InvGate's Blog, accessed June 4, 2025, https://blog.invgate.com/it-lifecycle-management
52. Project to Product Transformation Case Study: Global Media Service Provider, accessed June 4, 2025, https://itrevolution.com/articles/project-to-product-transformation-case-study-global-media-service-provider/
53. How do successful startups transition from cloud to hybrid infrastructure as they scale?, accessed June 4, 2025, https://www.quora.com/How-do-successful-startups-transition-from-cloud-to-hybrid-infrastructure-as-they-scale?top_ans=203061988
54. Maximizing Efficiency: Cloud Services and On-Premise Integration - Developers.Dev, accessed June 4, 2025, https://www.developers.dev/tech-talk/integrating-cloud-services-with-on-premise.html
55. Cloud vs. On-Premise: Which Is Better for Modernizing Business Operations?, accessed June 4, 2025,

https://imaginovation.net/blog/cloud-vs-on-premise-modernizing-business-operations/
56. 10 strategies and tactics for It Infrastructure Team - Tability, accessed June 4, 2025, https://www.tability.io/templates/strategies/tags/it-infrastructure-team
57. Transformative Case Studies of Companies Leading the IT Revolution through Infrastructure as Code Practices - MoldStud, accessed June 4, 2025, https://moldstud.com/articles/p-transformative-case-studies-of-companies-leading-the-it-revolution-through-infrastructure-as-code-practices
58. IT Infrastructure Automation & Orchestration with Itential, accessed June 4, 2025, https://www.itential.com/solutions/it-infrastructure-automation-orchestration/
59. Definition of Operating Model - Gartner Information Technology Glossary, accessed June 4, 2025, https://www.gartner.com/en/information-technology/glossary/operating-model
60. Best practices to work with a platform engineering team - Port IO, accessed June 4, 2025, https://www.port.io/blog/work-with-platform-engineering-team
61. Creating High-Impact Platform Teams for Better Software Delivery - Maruti Techlabs, accessed June 4, 2025, https://marutitech.com/platform-engineering-best-practices/
62. Tier 1, tier 2, tier 3: how to structure a tech support team for SaaS - ALLSTARSIT, accessed June 4, 2025, https://www.allstarsit.com/blog/tier-1-tier-2-tier-3-how-to-structure-a-tech-support-team-for-saas
63. IT Support Levels: How L0, L1, L2, L3, L4 Support Tiers Work - Desk365, accessed June 4, 2025, https://www.desk365.io/blog/it-support-levels/
64. Overview of Tier1/ Tier2/ Tier3 IT Support Engineers - Ecosmob, accessed June 4, 2025, https://www.ecosmob.com/overview-of-tier1-tier2-tier3-it-support-engineers/
65. Tiered Support Explained - HDI, accessed June 4, 2025, https://www.thinkhdi.com/library/supportworld/2018/tiered-support-explained
66. 10 Ways to Organize Your Product Team Structure, accessed June 4, 2025, https://productschool.com/blog/product-strategy/product-team-structure
67. Why Knowledge Workers and Enterprises Are Shifting to On-Premise GenAI - E-SPIN Group, accessed June 4, 2025, https://www.e-spincorp.com/shifting-to-on-premise-genai/
68. Tier 3 Help Desk | Level 3 Support Launch Plan ☑️ - ScienceSoft, accessed June 4, 2025, https://www.scnsoft.com/it-operations/help-desk/tier-3
69. Leveraging IT Support Models for Business Continuity - Spinnaker Support, accessed June 4, 2025, https://www.spinnakersupport.com/blog/2023/12/11/support-model/
70. DevOps Support Services | Geniusee, accessed June 4, 2025, https://geniusee.com/devops-support
71. Product Mindset as an Imperative – DASA, accessed June 4, 2025, https://www.dasa.org/blog/product-mindset-as-an-imperative/
72. Software Development with a Product Mindset - Oteemo, accessed June 4, 2025, https://oteemo.com/blog/software-development-with-a-product-mindset/

73. Breaking Down Silos: 10 Strategies for Cross-Functional Collaboration - Exclaimer, accessed June 4, 2025, https://exclaimer.com/blog/10-key-strategies-for-cross-functional-collaboration/
74. Integrating IT Talent Into Business Teams: Key Things You Should Know - Sellbery, accessed June 4, 2025, https://sellbery.com/blog/integrating-it-talent-into-business-teams-key-things-you-should-know/
75. AI-driven IT transformation for a Fortune 500 retail leader | HCLTech, accessed June 4, 2025, https://www.hcltech.com/case-study/breaking-the-it-firefighting-cycle-for-a-fortune-500-retail-leader
76. Infrastructure Engineer vs. Platform Engineer: Navigating the DevOps Landscape - Yardstick, accessed June 4, 2025, https://www.yardstick.team/compare-roles/infrastructure-engineer-vs-platform-engineer-navigating-the-devops-landscape
77. Transition from SysAdmin to Devops what does it require - Reddit, accessed June 4, 2025, https://www.reddit.com/r/devops/comments/1ijbwdq/transition_from_sysadmin_to_devops_what_does_it/
78. D&AI | Fortune Case Study - Accenture, accessed June 4, 2025, https://www.accenture.com/bg-en/case-studies/data-ai/fortune-turning-data-into-insights
79. The Essential Role of a DevOps Product Owner - DASA, accessed June 4, 2025, https://www.dasa.org/blog/the-essential-role-of-a-devops-product-owner/
80. IT Infrastructure Management: Strategies & Best Practices - Atlassian, accessed June 4, 2025, https://www.atlassian.com/itsm/it-operations/it-infrastructure-management
81. Digitally transforming Microsoft: Our IT journey - Inside Track Blog, accessed June 4, 2025, https://www.microsoft.com/insidetrack/blog/digitally-transforming-microsoft-our-it-journey/
82. IT Onboarding: A Comprehensive Guide for Admins - JumpCloud, accessed June 4, 2025, https://jumpcloud.com/blog/it-onboarding-guide
83. What is ITIL? A Complete Guide to the IT Infrastructure Library - LogMeIn, accessed June 4, 2025, https://www.logmein.com/resources/what-is-itil-complete-guide
84. The Complete Guide to Managing Hybrid Teams: Balancing Local and Outsourced Development - Full Scale, accessed June 4, 2025, https://fullscale.io/blog/hybrid-team-management-framework/
85. Principles of Infrastructure: Case Studies and Best Practices - World Bank PPP, accessed June 4, 2025, https://ppp.worldbank.org/sites/default/files/2022-03/adbi-principles-infrastructure-case-studies-best-practices.pdf