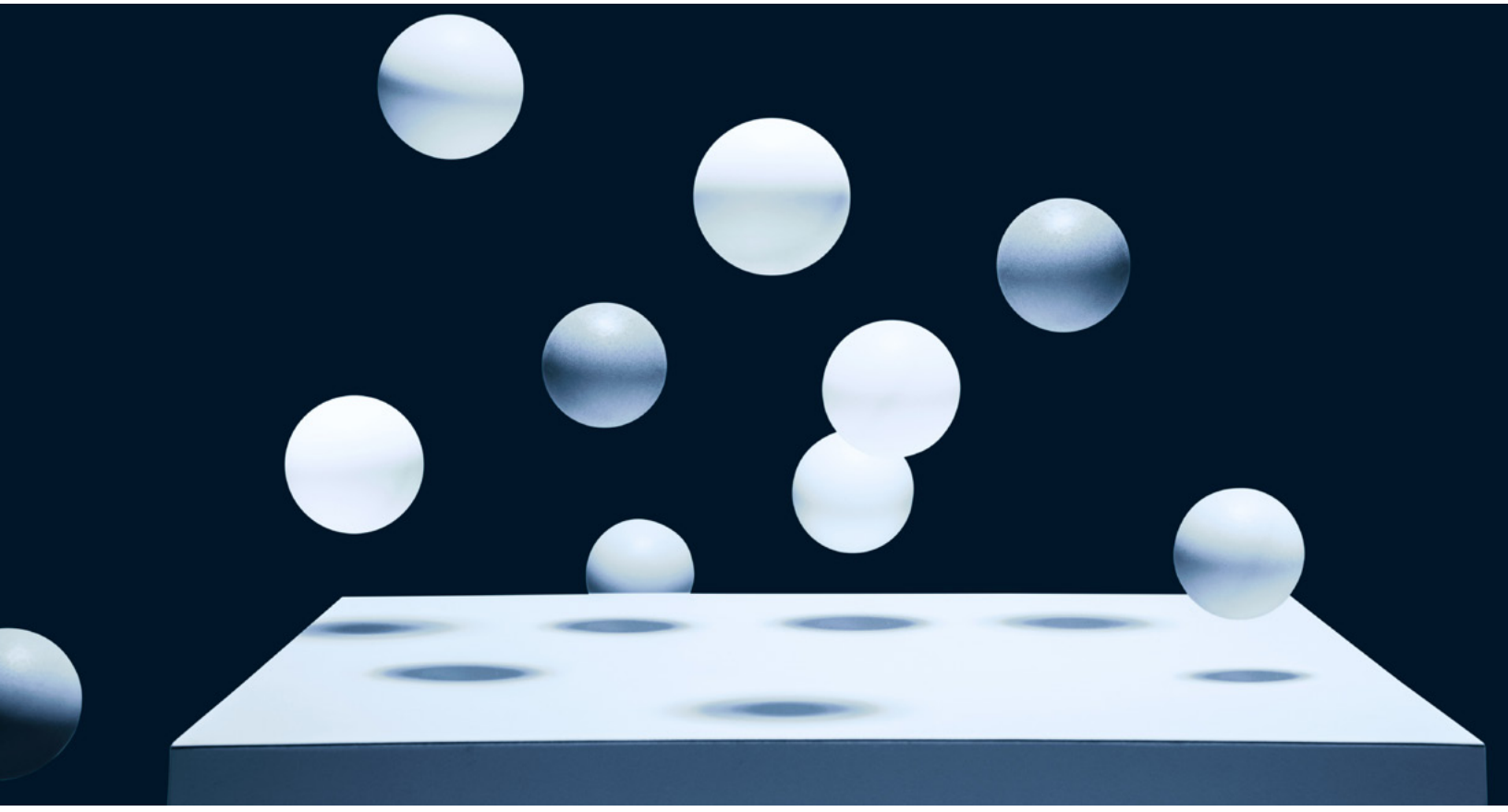
McKinsey Digital

Building a cloud-ready operating model for agility and resiliency

Four operating-model changes can help companies accelerate the journey to cloud.

This article was a collaborative effort by Santiago Comella-Dorda, Mishal Desai, Arun Gundurao, Krish Krishnakanthan, and Selim Sulos, representing views from McKinsey Technology.



March 2021

With customer expectations and technology evolving at an unprecedented clip, moving to cloud is increasingly becoming a strategic priority for businesses. Capturing the $1 trillion value up for grabs in the cloud, however, has proven frustratingly difficult for many companies. One of the main reasons for this difficulty is that IT's operating model remains stuck in a quagmire of legacy processes, methodologies, and technologies.

Overcoming this problem requires business and IT to take a step back and think holistically about their cloud operating model. And they need to move now. IT has become integral to driving value and a crucial enabler in meeting business and customer expectations of speed, flexibility, cost, and reliability. At the same time, the risk of failure is increasing because of the growth in complexities and demands around new architectures, agile application development, on-demand access to infrastructure through self-service, cloud migration, and distributed computing, to name a few.

While most organizations will need to adopt a hybrid-cloud approach for the foreseeable future, it will be hard to capture much of cloud's value without reimagining the IT infrastructure that is ground zero of the cloud operating model. Set up correctly, infrastructure can quickly expand access to new services and products, accelerate time to market for application teams, and cut operating costs at the same time—all of which unleash businesses' innovation potential.

To capture these benefits, companies must undertake a holistic transformation of infrastructure grounded on four mutually reinforcing shifts: adopt a site-reliability-engineer (SRE) model,[1] design infrastructure services as products, manage outcomes versus activities, and build an engineering-focused talent model. The benefits of these shifts can accrue to infrastructure and operations (I&O) even if they remain completely on-premises.

While each of these practices is well established, not many IT organizations have so far brought them together into a single operating model where they can function as a powerful combination of capabilities that can radically improve how IT operates. But we have seen companies that do so simultaneously improve resiliency, labor productivity, and time to market by 20 percent or more. One B2B service provider that carried out this transformation experienced a 60 percent reduction in change failure rate (the rate or frequency with which a system or service fails) while reducing labor spend by 30 percent.

Perhaps more important than understanding the power in combining these four elements of the model is having a practical approach to making it happen, derived from lessons in the field. The complexity and scale of an infrastructure transformation make evident the value of careful orchestration, creating points of integration with a wide array of functions across IT and the business, and sequencing activities to reduce risk.

Digging into infrastructure challenges

To adopt a cloud-ready IT operating model, companies need to address four challenges:

Legacies of manual intervention. Many companies still function on a manual-intervention-based operating model. Manually performing such tasks as submitting a ticket to make an update or offering multiple service catalogs to different departments[2] hurts application reliability and slows time to market. In fact, one of our clients had more than 300 I&O professionals implementing changes to production and pre-production environments. Detailed analysis of their critical incidents revealed that around one-third of outages were caused by human error. The issue wasn't so much a lack of rigor as it was a matter of statistics. No matter how many checks and balances there are, human interventions cause errors.

[1] An SRE model incorporates aspects of software engineering, such as capacity planning, and applies them to infrastructure problems. The goal of this model is to enable scalability and improve the reliability of software systems.

[2] Multiple service catalogs are the array of IT services that can be performed for different functions, such as HR, administrative, or finance. An example of a service category would be software, and a service could be software distribution for HR.

Lack of ownership clarity. Fragmented lines of responsibility create confusion about who should be doing which tasks. It's not uncommon, for example, to find dozens of IT infrastructure specialists on a production-incident resolution call because no one is certain who owns the task. As a result, customers can experience delays in restoration of infrastructure services or in a product release.

Misaligned success metrics. Service-level metrics are traditionally defined by activities or individual team outputs, and funding is based on volume. This creates an incentive to increase the amount of activity, rather than improve performance.

Too much operations, too little engineering. Typically, in many I&O departments, as many as six in ten people—system administrators, first-level support and monitoring technicians, and second-level infrastructure specialists who fulfill service requests—focus on operations, while fewer than three in ten focus on engineering new capabilities, and the balance provide managerial support.

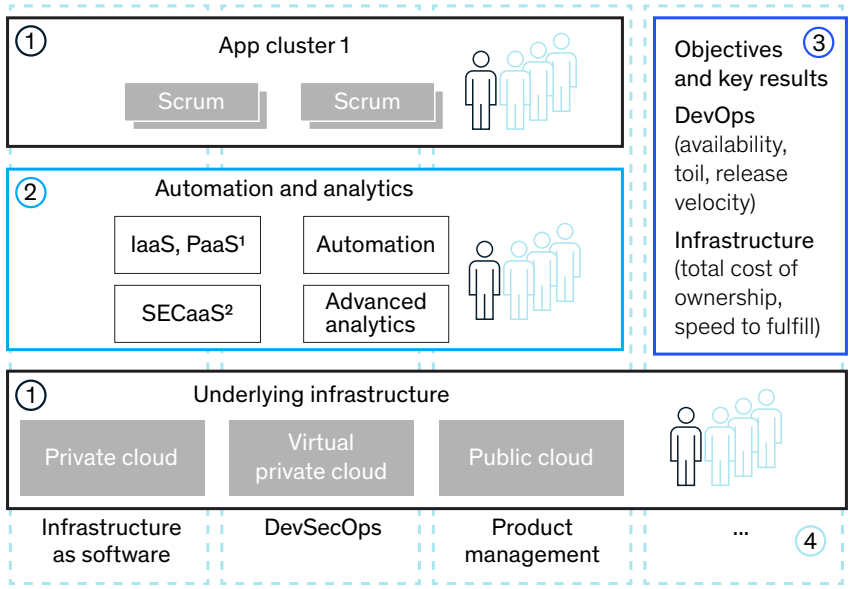
Often the ratios are even worse. One banking institution found that for every dollar it invested in engineering public and private cloud, it was investing $2.50 in operations. Exacerbating this issue is that workers with specific role profiles often lack experience across legacy and cloud environments. For example, many organizations have database administrators who specialize in a single database, such as Oracle, SQL Server, or PostgreSQL. In fact, we've found that many professionals can master full classes of databases, such as relational or non-SQL, across different cloud environments.

Build a new IT-infrastructure operating model

To build out a cloud-ready operating model, IT leadership should focus on four actions (see exhibit). How companies go about implementing this model will vary, of course, depending on their specific situation and goals (see sidebar, "Questions to help tailor the model").

Exhibit

A cloud-ready operating model includes four key components.



1 **DevOps/site reliability engineering**
Small teams responsible for resiliency, "toil" reduction, and DevSecOps adoption

2 **Productized infrastructure services**
Cross-functional teams responsible for building and maintaining discrete products

3 **Outcome-driven governance**
Delivered through agile ways of working

4 **Engineering-centric capabilities**
Focused on building world-class engineering talent and minimizing low-value operations

¹IaaS = infrastructure as a service; PaaS = platform as a service.
²SECaaS = security as a service.

Questions to help tailor the model

The model described in this article needs to be tailored to the specific situation and goals of each institution. An organization that is aggressively migrating to cloud, for example, will focus on public-cloud-only services and design SRE teams to assist application development on the migration. An organization that plans to continue relying on a private-cloud or on-premises infrastructure model, on the other hand, will maximize the number of infrastructure products that can work seamlessly across environments and set up SRE teams that can eliminate existing toil (manual or low-value-added work).

The following questions provide important guidance to tailoring the infrastructure operating model:

Structure and composition of the SRE function

— What should be the balance between "application SREs" (attached to a particular app or apps) versus "solution SREs" (developing application services and tools)?

— Should the SREs be mostly a consultative function, and hence smaller, or an engineering and operational function, and hence larger?

— Should the SRE function stay in the infrastructure organization, or should it be moved into application development?

Structure and composition of the product infrastructure teams

— What should the product taxonomy and team alignment be across infrastructure as a service (IaaS), platform as a service (PaaS), and service management?

— How should level-one support and operations, such as monitoring and incident management, be organized?

— For each product family, should engineering and operations be handled by the same team or separate teams?

— Which product services and teams should be multicloud—working across both private and public cloud—versus cloud specific?

— Should the infrastructure product structure be reflected in the organizational chart, or should it be formed with virtual teams?

Objectives and key results (OKRs) and metrics

— What specific metrics should a product team and SRE teams track, particularly around business outcomes?

— How should the OKRs cascade and reflect, in particular, the balance between top-down and bottom-up objectives?

— How should OKRs relate to team- and personal-performance management, if at all?

Engineering-centric talent management

— What should be the balance between developing the existing workforce and acquiring new talent?

— To what extent should career paths and job descriptions be redesigned, broadened, and merged?

— How can you effectively work with external service providers, particularly on those cases in which large fractions of I&O have been outsourced? Should the organization insource some of those functions?

Adopt a site-reliability-engineer model

Site reliability engineers (SREs) are the glue that binds application development and core infrastructure services. They work cross-functionally, partnering with application developers, application operations, and infrastructure teams. They also enhance the stability and reliability of applications in production and reduce or automate repetitive manual tasks so that the development team can focus on building products. To help applications run uniformly and consistently on any cloud infrastructure, SREs should also support containerization and replatforming efforts.

In general, however, the most effective SREs maintain a balance of responsibilities between engineering and operations, with a greater than 50 percent emphasis on engineering.

On the operations side, SREs oversee incident management, keep track of service-level objectives and indicators, coordinate product releases, and perform such hands-on tasks as monitoring systems and resolving incidents. They also drive the adoption of DevSecOps, a method of integrating security into the agile and DevOps processes, which helps to increase the frequency of software releases from quarterly to weekly or even daily, cut mean time to remediate vulnerabilities from weeks or months to hours, and eliminate delays, cost overruns, product defects, and vulnerabilities.

On the engineering side, SREs manage the instrumentation (monitoring infrastructure) and improve the reliability and scalability of services through automation by creating, for example, a continuous-integration and continuous-delivery (CI/CD) pipeline for infrastructure and application components; reduce technical debt; and support capacity planning. As IT migrates to the new operating model, SREs may focus 70 percent or more of their time on operations. But as they remove "toil"—the manual or low-value-added work—and system complexity, the ratio will become more balanced; over time, SRE teams should spend most of their time on engineering activities.

Organizations can start by aligning SRE teams with applications or application clusters. As the organization's operating-model maturity increases and operations become automated, SREs can be embedded into the application-development teams. In some very mature teams with more homogenous technical stacks, site reliability can become the responsibility of full-stack engineers instead of being designated to a separate role. As this staggered approach increasingly blurs the lines between application development and infrastructure, organizations can more nearly approach the operating model of "hyperscalers" (large cloud and web providers with extremely efficient infrastructure).

The SRE model delivers rapid benefits because it brings infrastructure expertise closer to the applications and allows for direct, face-to-face collaboration across application development and infrastructure. That is a substantial shift for most organizations, in which infrastructure resources are pooled so that functional specialists serve the entire application portfolio.

Design infrastructure services as products

Silos make traditional infrastructure operating models incompatible with agile and cloud-ready infrastructure. For I&O to be responsive and fast, it must be organized based on the infrastructure products it supports rather than by roles.[3] To do so, companies must build agile product teams made up of people with relevant areas of expertise, including product owners, solution architects, infrastructure and software engineers, and security specialists. Product owners can collaborate with application teams to understand what services or products are needed. They should work with SRE teams to understand the challenges in consuming these services. This process helps prevent infrastructure teams from developing solutions that no one needs. The team can offer a catalog of existing services and a road map of upcoming services or improvements.

These agile product teams are responsible for end-to-end delivery and automation of discrete products to be consumed consistently, whether deployed in a data center or in a public cloud. These teams can also provide full life-cycle, self-service infrastructure assets, such as virtual servers or storage capacity, that can be set up, maintained,

[3]Ross Frazier, Naufal Khan, Gautam Lunawat, and Amit Rahul, "Products and platforms: Is your technology operating model ready?" February 2020, McKinsey.com.

and taken apart, all through automated services. Such teams can either be accountable for the operations associated with the assets or collaborate with the operations team that supports them, often through Kanban.[4] For example, a product team could offer Linux (the open-source operating system) as a service and publish application programming interfaces (APIs) to enable setting up, taking apart, and patching infrastructure assets, among other associated services.

Manage outcomes versus activities

Setting objectives and key results (OKRs) at the outset of the transformation helps application development and infrastructure teams align on what they want to achieve with their new, agile, automated IT infrastructure. These metrics also create accountability across teams. Historically, most organizations either focus on tracking activities or have different OKRs for different teams, which is why many miss out on potential value. While these sorts of metrics have their place, hybrid-ready infrastructure teams need to be measured against business outcomes, such as customer adoption.

Three components are especially critical:

Build goals from the top down and bottom up. Most organizations rely exclusively on dictating business goals from the executive level down. In contrast, successful organizations combine those goals with a healthy mix of team-level goals channeled from the bottom up. A frontline infrastructure engineer often has the best understanding of resiliency issues that could bring down operations.

Set tangible stretch goals. Goals should be hard enough to promote outside-the-box thinking and present a challenge. In fact, teams likely will not hit all the goals they set. If a team consistently meets every goal, more likely than not, the goals were not hard enough to start with. Setting stretch goals and targets gets teams excited about transformation and can create a culture of collaboration. Goals also need to be tangible and specific. Metrics must be detailed enough to be traced back to a team, tracked quarterly, and attributed to a business outcome so that executives can see where they need more coverage or support.

Have the right tools and support to measure outcomes. Teams need access to granular and recent metrics on reliability (availability and failure rates), response time (milliseconds per transaction), and cost (percentage of toil and total cost of ownership). To get this level of insight, teams need CFO support and access to tools that can track financial and nonfinancial targets. Then teams can tie targets to specific tasks and product features to understand their performance and how they add value.

The magnitude and complexity of building the new IT infrastructure operating model requires careful orchestration and coordination. As such, companies often set up a transformation office (TO), led by a senior executive, such as the chief information officer or the head of infrastructure, to steer the effort. While the TO should include the standard functions of a good project-management office—such as setting goals and boundaries, planning, and tracking progress—it must reflect the greater scope of the effort. That means, for instance, working closely with HR to hire the right talent, collaborating with developers and business sponsors to deliver outcomes, and bringing in people with sufficient domain expertise to manage complex decisions.

Build an engineering-focused talent model

As companies move away from manual solutions, they will need to build a bench of engineering talent that can develop automated infrastructure solutions, such as an automated, self-healing, virtual machine that can find errors or malfunctions and solve them independently.

Finding people with critical skills and knowledge about processes and models such as infrastructure as code, SRE, and product ownership often requires organizations to pursue internal capability building and hire external talent. Organizations can build capabilities through formal/classroom training and—most important—informal mentoring and apprenticeship with senior engineers. Techniques such as peer and code reviews, in which senior engineers review and provide feedback on the work of their peers, can support knowledge transfer and establish consistent levels of experience.

[4]Kanban is a lean operating model that helps manage and improve work across teams.

Capability building can also help engineers gain depth of knowledge in critical areas such as code as software and build up breadth of knowledge by cross-training engineers (such as storage specialists learning compute cloud skills). In many cases, engineers have managers at multiple levels who need to review work, which generally slows the engineering process down. Instead, organizations should give their best engineers the freedom to work quickly, which often means giving them space within a set of fixed guardrails to develop their own solutions.

Case example: Putting everything together

A large Asian bank faced competition from other digital banks and fintechs powered by agile cloud-based platforms. The bank knew it needed to migrate to a cloud-ready infrastructure so that it could quickly develop and test new products, such as a mobile-payment system. However, legacy IT infrastructure; siloed teams across network, database, and storage computing functions; and manual ticket-based workflows made it nearly impossible to modernize its I&O for cloud.

At the outset of the transformation, the bank set clear and aspirational OKR targets to reduce hard, manual, repetitive work; to improve efficiency; and to automate new product testing. It also restructured how teams work together, stressing the importance of collaboration on application development and infrastructure.

Setting these goals also helped the bank avoid the common pitfall of starting a transformation without focus, which can result in operating-model changes that increase IT complexity. Counterintuitively, setting high goals helped bank employees feel freer to test, learn, and think creatively to achieve their targets.

To make progress against its OKRs, leadership rebalanced the ratio of engineering and operations talent and focused on a targeted set of productized infrastructure offerings with support from SRE teams to reduce toil and dramatically improve stability. To get the right engineering talent on board, the bank first looked internally and found that 80 percent of its engineers could be reskilled and moved into different or new roles. This sensitivity to culture and prioritization of internal talent also helped the bank get its teams on board with the transformation. Shifting to an engineering-focused organization and gaining the support of advanced engineers also made it possible to adopt an SRE model.

The bank orchestrated broad adoption of its new operating-model approaches across all application teams; as a result, more than 90 percent of its applications were able to run on productized infrastructure. In addition, because SRE teams focused on toil elimination and had tight control over productized infrastructure, the bank increased its ratio of operating-system images to headcount by 50 to 100 times.

In the end, the bank successfully rolled out a new mobile-only banking offering in its target geography at unprecedented speed. Its new operating model helped the organization slash deployment time from weeks to hours, increase delivery cadence by ten times, and reduce operating costs by 30 percent (50 percent compared with revenue), while still delivering six times the number of operating-system instances.

Building an IT infrastructure operating model for the future is a complex endeavor, but it is essential for companies that want to survive and thrive at the pace of digital.

Santiago Comella-Dorda is a partner in McKinsey's Boston office, where **Mishal Desai** is a consultant; **Arun Gundurao** and **Selim Sulos** are associate partners in the New York office; and **Krish Krishnakanthan** is a senior partner in the Stamford office.

The authors wish to thank Nagendra Bommadevara, Thomas Delaet, Andrea Del Miglio, Vito Di Leo, Mark Gu, Steve Jensen, James Kaplan, Ling Lau, Pablo Prieto-Munoz, and Kalin Stamenov for their contributions to this article.