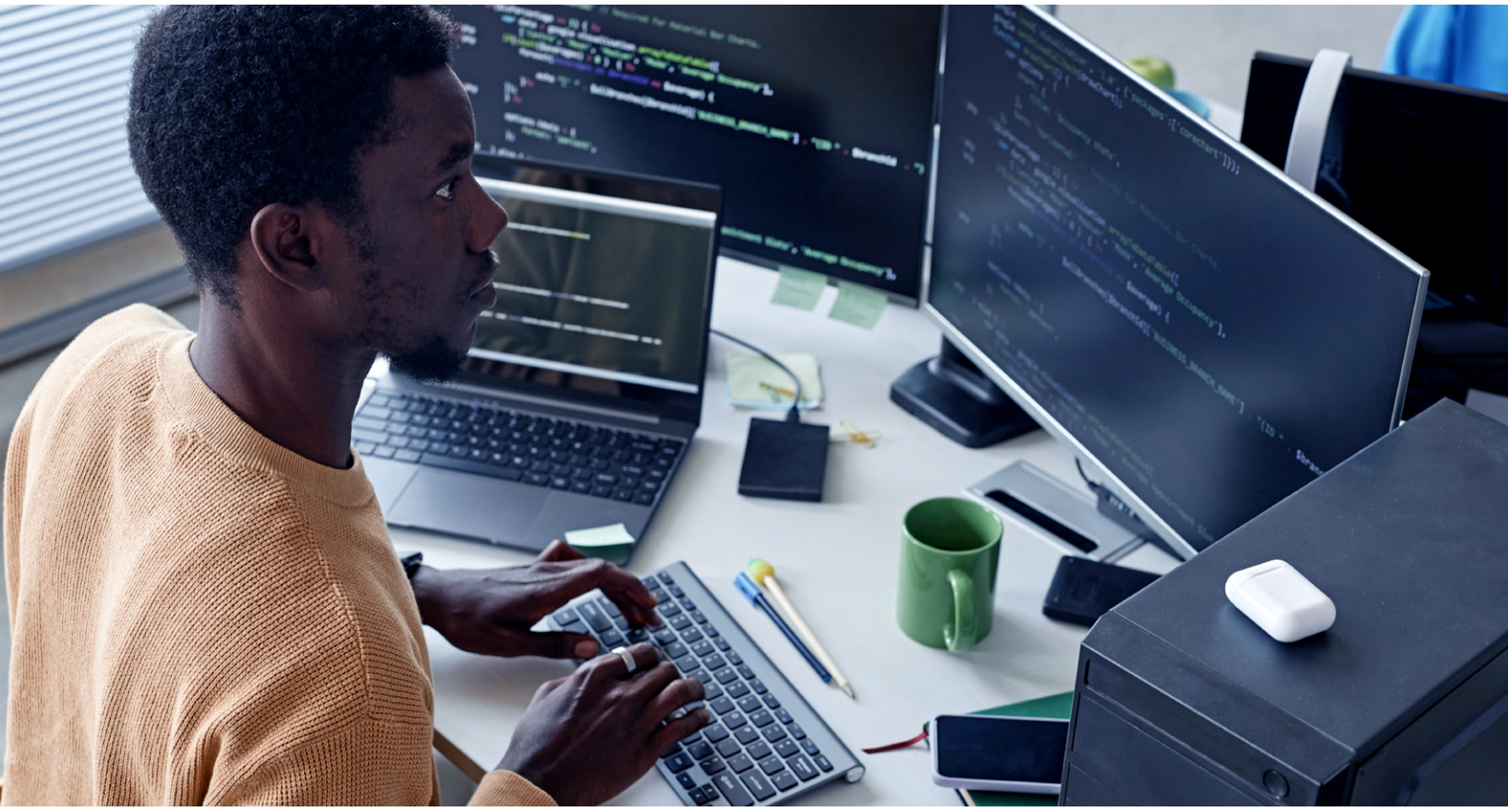
McKinsey Digital

Everything is better as code: Using FinOps to manage cloud costs

Turning cloud cost principles and policies into code that's integrated into the engineering process could generate almost $120 billion in value.

This article is a collaborative effort by Abhi Bhatnagar and Pablo Prieto, with Ankit Saraf, Bailey Caldwell, Konstantin Tyrman, and Zubin Ghafari, representing views from McKinsey Digital.



February 2025

Does this sound familiar? A company spends more than $50 million a year on cloud services with a small FinOps team dedicated to tracking expenses. This team, supported by contract labor, uses various solutions to monitor and control spending and to identify inefficiencies and opportunities for improvement. Yet cloud waste persists, from storing unnecessary data to using overly expensive services.

Such situations are not unique. We looked at more than $3 billion in cloud spending across organizations and industries and found that most organizations had additional untapped cost savings of 10 to 20 percent (see sidebar, "Methodology"). But even when a significant opportunity is identified, FinOps teams often find it difficult to capture savings beyond their mandate. One major reason is that engineers don't have the incentives or access to act on cloud costs.[1] Actions to optimize costs often fall by the wayside for engineers already stretched thin by working on several business priorities, implementing resiliency actions, and improving security.

That's why some organizations are adopting an "everything as code" philosophy for FinOps, similar to the approach with security and infrastructure.[2] FinOps as code (FaC) helps reduce costs by automatically integrating FinOps best practices into engineers' workflows. When combined with observability techniques and a solid set of policy guardrails, FaC can optimize costs.

We estimate the potential value from FaC to be about $120 billion, based on expected spending of roughly $440 billion on global cloud infrastructure as a service (IaaS) and platform as a service (PaaS) in 2025[3] as well as the roughly 28 percent of cloud spending that organizations report as waste.[4]

In this article, we examine the role of FaC when integrated into the infrastructure management life cycle (IMLC), the benefits for engineering teams, and a toolkit that organizations can use to make FaC a reality.

What is FaC?

FaC is a practical approach to integrating financial management principles into the IMLC[5] to automatically manage cloud costs. By using a combination of automation, policy enforcement, and cloud-native services, FaC enables organizations to implement FinOps guidelines directly into development, deployment, and infrastructure provisioning pipelines[6] (exhibit). In addition, FaC can be used to enforce budgets and cost-efficient architecture practices, automatically identify areas of cost reduction, and support better resource scheduling.

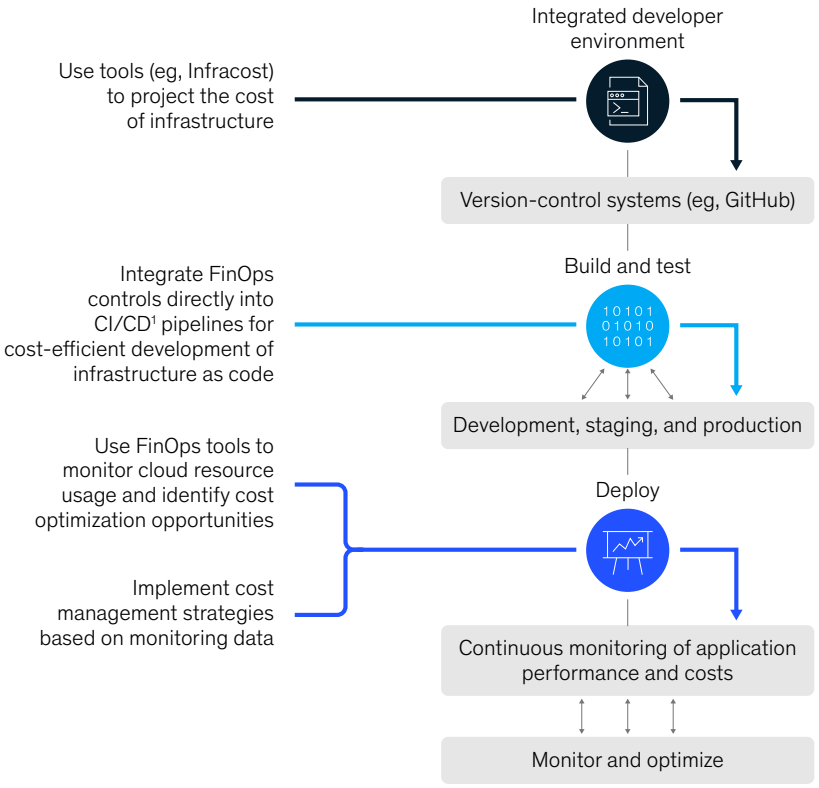
1 "Top challenges for practitioners shift and evolve in 2023," FinOps Foundation, 2023.
2 "Security as code: The best (and maybe only) path to securing cloud applications and systems," McKinsey, July 22, 2021.
3 "Gartner forecasts worldwide public cloud end-user spending to surpass $675 billion in 2024," Gartner, May 20, 2024.
4 Tanner Luxner, "Cloud computing trends and statistics: Flexera 2023 state of the cloud report," Flexera, April 5, 2023.
5 IMLC refers to the stages involved in planning, implementing, operating, and maintaining an organization's IT infrastructure.
6 An infrastructure provisioning pipeline is the automated processes and tools used to deploy and manage cloud infrastructure using infrastructure-as-code principles.

Methodology

We undertook a comprehensive review of detailed cloud bills using a proprietary set of algorithms to analyze consumption patterns and spending to identify savings opportunities. The output was then paired with other ancillary inputs, interviews with key cloud stakeholders, and, if available, detailed resource consumption data from monitoring and observability software. The quantitative and qualitative data provided a robust and holistic picture of the cloud portfolio.

Exhibit

FinOps as code enables FinOps guidelines to be coded directly into infrastructure provisioning pipelines.



[1]Continuous integration and continuous deployment.

McKinsey & Company

If FaC is properly integrated into the IMLC, FinOps reduces architectural debt (the complex or one-off coding configurations that make future adjustments more time consuming and expensive) and strengthens the quality of the code. It also decreases the need for manual intervention by using code and automation to enforce cloud cost governance policies.

These new FaC policies can also be applied across a wide array of granular cloud assets or accounts to optimize their usage and achieve cost savings. For example, a large retailer converted utilization metrics into FaC rules that identified opportunities to shut down servers on nights and weekends. As a result, the retailer reduced its cloud costs by approximately 6 percent. This approach can also help avoid potential waste in new growth areas, ensuring efficiency from the start.

The benefits of FaC for engineers

Compared with a traditional, more manual approach to FinOps, FaC presents several benefits for the engineering teams:

— *Automating cost optimization.* Cloud providers are continuously introducing new cloud services that may be more cost efficient and

Rolling out FaC does not have to be overly burdensome despite its apparent complexity.

perform more effectively. FaC lifts the burden of implementing these changes from engineers and uses code to incorporate changes automatically into their workflow. For example, a cloud provider introduced an optimized storage offering that was cheaper and more compatible for virtual machines (VMs).[7] Once FinOps teams render that offering into code and deploy it, legacy storage models can be upgraded to the new VMs automatically.

— *Providing real-time cost visibility and accountability for engineers.* FaC tooling provides engineers with immediate visibility into the cost implications of their designs in an integrated development environment[8] so they can make decisions that are more cost-effective before deploying code. This real-time feedback fosters a shift-left culture of financial accountability. As part of deployment processes, developers can review financial reports and organizations can enforce budget constraints.

— *Reducing disruption and "maintenance" work.* The traditional "annual spring cleaning" approach—the practice of periodically performing maintenance and cleanup activities on a codebase—often requires companies to take resources offline or make significant changes to configurations. Doing so disrupts work and often negatively affects customers, and it is not a good use of engineers' time. Instead, engineering teams could employ FaC to continuously optimize resources and configurations, thereby cutting back on disruptions and saving engineers' time for more important work.

— *Enabling more-efficient cloud resource allocation and planning.* Engineering teams can use FaC to identify in real time where specifically cloud resources are being used inefficiently and automate the remediation of those inefficiencies. Optimizing this process frees up operating budgets so funds can be reinvested into additional innovation and feature development.

— *Identifying forgotten or unused infrastructure.* Whether through rapid development and expansion of applications or as part of system retirement processes, infrastructure components such as unallocated IP addresses, network interfaces, backups, and snapshots may continue incurring cost for the organization. FaC can help identify and, through automated remediation processes, remove these components. This can also resolve charge-back gaps and visibility issues into costs associated with assets and applications that should have been retired.

The FaC toolkit

Rolling out FaC does not have to be overly burdensome despite its apparent complexity. It does require companies to clearly understand

[7] "Comparing Amazon EBS volume types gp2 and gp3," Amazon Web Services, accessed January 13, 2025.
[8] An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development.

what's needed and commit to incorporating FaC into production. Making FaC a reality demands organizations focus on four components:

— *Specialized tools.* Effective FaC implementation requires the right tools to enforce policies. FaC implementation requires tools (for example, Open Policy Agent) that validate infrastructure-as-code (IaC) scripts against predefined policies that are written in a declarative language optimized for policy setting (for example, Rego). Other tools, such as those that offer IaC guardrails similar to policy as code (PaC), target different areas of cloud management.

— *Policy categorization.* Organizations can establish a simple categorization of policies based on their function. One example is inform, warn, and block policies, which are a set of access control measures that manage user access to resources based on predefined conditions. Inform policies provide well-architected recommendations to the team, such as leveraging a cloud-native database service instead of a DIY approach. Warn policies alert teams—without preventing provisioning—about best practices being missed (for example, an autoscaling group that is not established). Block policies prevent deployment altogether if certain conditions are met, such as provisioning more than $100,000 worth of infrastructure for a development environment. Best practices include running these policies against IaC at every pull request before promoting to production.

— *Policy scripts.* Policy scripts are the backbone of FaC. Gen AI can simplify policy creation by using natural language prompts, training on a large data set of FinOps controls and policies, and implementing retrieval-augmented generation (RAG). Early in the FaC journey, organizations need only ten to 15 policies. To determine which policies to create, organizations can examine prominent sources of waste, such as maintaining nonproduction environments during nonworking hours or setting unnecessarily long log retention periods. Teams can then identify and test a target use case that can be implemented with limited resources.

— *Change management.* Successful FaC implementation requires effective change management. As policies are rolled out, it is crucial to notify all key stakeholders of new enforcement policies and to ensure engineers understand what FaC is, what its value is, and how it affects their day-to-day operations. This ensures smooth adoption and minimizes resistance to new processes.

Managing cloud costs should be more science and less art. Through FaC practices, companies can harness code for a range of benefits, from automating resource optimization to fostering financial accountability and reducing the strain on engineering teams to sustain optimization results.

Abhi Bhatnagar is a partner in McKinsey's Atlanta office; **Pablo Prieto** is a partner in the Connecticut office; **Ankit Saraf** is a principal delivery lead in the Chicago office; **Bailey Caldwell** is an associate partner in the Southern California office, where **Zubin Ghafari** is a principal architect; and **Konstantin Tyrman** is an associate partner in the New Jersey office.