International Journal of Computer Science Engineering Techniques – Volume 9 Issue 5, September - 2025

# FinOps-Aware DevOps Practices: Optimizing Cloud Cost Efficiency through Continuous Monitoring and Automation

Pruthvi Raj Seknametla

*Individual Researcher*

pruthviraj.seknametla@ieee.or, pruthviraj9369@gmail.com

***Abstract -*** *Cloud expenditure has become one of the fastest-growing line items in enterprise technology budgets, and the gap between what organizations provision and what they actually use is consistently larger than finance and engineering teams expect. FinOps the practice of bringing financial accountability to the variable-spend model of cloud infrastructure, has matured significantly since the FinOps Foundation formalized it as a discipline in 2019. Yet the integration of FinOps principles into active DevOps pipelines and engineering workflows remains poorly understood and inconsistently practiced. This paper examines how fifteen engineering organizations approached FinOps-aware DevOps across a twenty-month study period from February 2023 through September 2024. We analyze cloud cost anomaly detection rates, resource rightsizing adoption, idle and over-provisioned resource reduction, and the impact of cost-aware deployment automation on total cloud spend per engineering unit of output. Results demonstrate that organizations with mature FinOps-DevOps integration reduce cloud waste as a proportion of total cloud spend by an average of 47% compared to those managing cost through periodic manual review alone, and that continuous automated cost monitoring is the single highest-leverage intervention available to most organizations. We propose a four-pillar FinOps-DevOps integration model and document the organizational and technical barriers that most consistently limit effective implementation.*

***Keywords -*** *chargeback, cloud cost optimization, cloud unit economics, cloud waste reduction, continuous cost monitoring, DevOps automation, FinOps, Kubernetes cost management, resource lifecycle automation, rightsizing, showback, spot instances*

## 1. Introduction

There is a particular kind of quarterly business review meeting that engineering leaders in cloud-heavy organizations have come to dread. Finance pulls up the cloud bill. It has increased by 34% year-over-year. Engineering leaders explain that the company grew, so usage grew. Finance asks why costs grew faster than revenue. Engineering leaders talk about scaling events and new service launches. Finance asks about the $400,000 of EC2 instances that ran at 4% CPU utilization for six months. Nobody has a good answer, because nobody was watching.

This scenario is not hypothetical, and it is not uncommon. The Flexera 2024 State of the Cloud report found that organizations estimated they waste an average of 27% of their cloud spend resources provisioned but not meaningfully used. Among organizations with annual cloud bills above $10 million, that figure represents a material budget line that is effectively being discarded. Waste is not usually the result of negligence or incompetence. It is the result of a structural mismatch: cloud infrastructure is provisioned by engineers making technical decisions, but the cost implications of those decisions are reviewed by finance teams on a monthly or quarterly cycle, long after the decisions have been baked into production deployments and forgotten about.

FinOps short for Financial Operations emerged as a formal discipline to address exactly this mismatch. The FinOps Foundation, established in 2019, defines FinOps as a cultural practice that enables organizations to get maximum business value from cloud investment by bringing together engineering, finance, and business teams to make collaborative, data-driven spending decisions. The foundational insight is that cloud cost optimization is not a finance function or a DevOps function it is a cross-functional practice that works only when the people provisioning infrastructure understand its cost implications in real time, and when cost data is visible and actionable in the tools and workflows those people use.

The integration of FinOps into DevOps pipelines and practices is the operational expression of this insight. When cost data surfaces in pull request reviews, deployment pipelines, and Kubernetes workload dashboards rather than in monthly finance reports delivered after the fact engineers can make cost-informed decisions without context-switching to financial tooling. When automated policies can flag or prevent resource configurations that exceed cost thresholds, waste accumulation is interrupted before it compounds.



ISSN: 2455-135X https://www.ijcsejournal.org/

When cost is treated as a deployment metric alongside latency and error rate, it becomes part of the engineering team's operational awareness rather than a surprise at the end of the quarter.

This paper examines how fifteen organizations built FinOps-DevOps integration in practice, what they measured, what changed, and where they got stuck. The study period spans February 2023 through September 2024, a period during which cloud cost scrutiny intensified significantly as interest rate increases put pressure on technology company valuations and engineering budget efficiency became an executive priority across the industry.

### *1.1 The Cost Visibility Problem in DevOps*

Traditional IT cost management operated in a world of capital expenditure: servers were purchased, their depreciation was scheduled, and their cost was a fixed line item that finance tracked independently of engineering decisions. Cloud infrastructure inverted this model entirely. Every provisioning decision, every scaling event, every service call to a managed database or messaging queue generates a cost that accumulates in real time. A developer who launches an oversized RDS instance for a development environment and forgets to turn it off after the sprint ends is not making a budget decision in any conscious sense. But they are spending money, and without visibility into that spend, the decision cannot be corrected.

The problem is compounded in DevOps environments by the pace and volume of infrastructure change. An organization deploying multiple times per day to Kubernetes clusters across three cloud environments is generating hundreds of infrastructure provisioning events per week. The cost implications of each event are individually small and collectively significant. Manual cost review processes examining monthly bills, identifying anomalies, mapping charges back to specific teams and services cannot operate at the cadence those environments require.

Cost visibility also has an attribution dimension. Cloud bills aggregate charges at the account or project level by default. Disaggregating them to the team, service, or feature level requires deliberate tagging and labeling strategies that most organizations do not implement consistently. Without attribution, cost accountability is impossible: you cannot hold a team responsible for overspending if you cannot determine which spending was theirs.

### *1.2 Research Questions*

This study investigated four primary questions. First, which FinOps integration patterns produce the largest measurable reductions in cloud waste, and at what organizational and technical cost? Second, how does continuous automated cost monitoring compare to periodic manual review in terms of anomaly detection timeliness and waste reduction? Third, what role does resource lifecycle automation play in reducing idle and over-provisioned resource costs, and what operational risks does it introduce? Fourth, what organizational structures and incentive models distinguish engineering teams that actively optimize cloud costs from those that treat cost as someone else's problem?

## 2. Literature Review

### *2.1 The FinOps Foundation Framework*

The FinOps Foundation's framework organizes cloud financial management around three phases: Inform, Optimize, and Operate. The Inform phase focuses on visibility, understanding current spending, attributing it to teams and services, and establishing unit economics baselines. The Optimize phase focuses on identifying and acting on cost reduction opportunities, rightsizing, reserved instance commitments, idle resource elimination. The Operate phase focuses on making cost management a continuous organizational practice with clear ownership, defined processes, and regular review cadences.

The framework's emphasis on organizational practice rather than tooling is deliberate and important. Castellanos and Harmon (2022) analyzed FinOps adoption patterns across enterprise cloud organizations and found that the most common failure mode was organizations that invested heavily in cost monitoring tooling but did not change their organizational processes cost data became more visible without becoming actionable, because no team had clear ownership of acting on it. The Inform phase without the Operate phase produces well-instrumented waste.

Kelly (2023), in her analysis of cloud cost management maturity across technology-intensive industries, identified what she termed the 'FinOps handoff problem': cost optimization insights generated by centralized cloud infrastructure teams were not being consumed or acted upon by the application engineering teams who had the authority to change the resource configurations driving the costs. The organizational gap between the team that sees the cost data and the team that can change the spend is the root cause of waste accumulation in most mature cloud organizations.

### *2.2 Cloud Waste: What It Is and Where It Hides*

Cloud waste exists in several distinct categories that require different detection and remediation approaches. Idle resources compute instances, databases, load balancers, and other infrastructure that is provisioned but generating no useful traffic or workload are the most visible category. Automated detection through utilization metrics is straightforward in principle, though threshold-setting (what utilization percentage constitutes 'idle'?) requires judgment about workload patterns.

Over-provisioned resources are subtler. An RDS instance sized for peak-period load running at 8% average



ISSN: 2455-135X https://www.ijcsejournal.org/

CPU utilization during normal operations is not idle it is serving real traffic but it is consuming significantly more than it needs to. Rightsizing over-provisioned resources is one of the highest-ROI optimization activities available, with McKinsey estimating that rightsizing alone can reduce compute costs by 15-25% in organizations that have not previously addressed it.

Orphaned resources are a particularly insidious waste category: storage volumes detached from deleted instances, static IP addresses reserved but not associated with any resource, snapshots accumulating without expiration policies, DNS entries pointing at decommissioned services. These resources generate costs indefinitely without providing any operational value.

Environmental sprawl development, staging, and testing environments that continue running continuously when they are only needed intermittently is a cost category that engineering teams often defend on operational grounds but which typically represents significant waste when workload patterns are examined. A development environment that is actively used eight hours per day, five days per week, needs to run for approximately 24% of the hours it might otherwise run continuously.

### *2.3 Kubernetes Cost Management: A Special Complexity*

Kubernetes introduced a layer of cost management complexity that cloud-native cost tools have only partially addressed. In a traditional VM-based deployment, each workload's cloud cost is relatively direct: the VM running the workload has a known instance type and cost. In Kubernetes, workloads share nodes, requests and limits define resource reservations that may or may not match actual consumption, and the relationship between a running Pod and its fraction of the underlying node cost requires attribution logic that most cloud billing APIs do not provide natively.

Kubecost, OpenCost (the open-source derivative donated to the CNCF by Kubecost), and commercial equivalents were all developed to address this gap by performing in-cluster cost attribution based on Pod resource requests and actual node costs. The OpenCost specification, published in 2022, defines a standard for Kubernetes cost allocation that multiple tools now implement.

The Kubernetes efficiency ratio the proportion of reserved resource capacity that is actually consumed is a critical metric for Kubernetes cost management. Organizations frequently set resource requests conservatively and set limits liberally, resulting in nodes that are heavily reserved but lightly consumed. A cluster reporting 80% resource reservation may have 30% actual CPU utilization, meaning 50% of reserved capacity is allocated but unused.

### *2.4 Cost-Aware Infrastructure as Code*

The idea of surfacing cost estimates for infrastructure changes before they are applied analogous to how a code review surfaces logic changes before they are deployed has gained traction with the development of tools like Infracost, which integrates with Terraform and other IaC tools to generate cost diff estimates for pull requests.

Infracost's research (2023) found that engineering teams with cost estimates in pull request workflows reduced infrastructure cost-related incidents by 43% compared to control groups without pull request cost visibility. The mechanism is simple: engineers who see that a proposed change would add a meaningful cost have the opportunity to evaluate whether that cost is justified and to consider alternatives.

Gruntwork and Banzai Cloud's broader work on 'infrastructure cost governance' has extended this concept toward policy-as-code for cost controls: rules that prevent infrastructure configurations above certain cost thresholds from being deployed without explicit approval.

## 3. Methodology and Proposed Model

### *3.1 Study Design and Participant Selection*

Fifteen organizations participated in the study, observed from February 2023 through September 2024, a period of twenty months. Participants were recruited through FinOps Foundation community networks, cloud platform engineering conferences, and referrals from cloud cost management consultancies. All fifteen had annual cloud bills exceeding $500,000. All fifteen provided anonymized billing telemetry, cost monitoring tool exports, and resource utilization data under data-sharing agreements.

Annual cloud spends at study entry ranged from $800,000 to approximately $47 million. Engineering headcounts ranged from 80 to approximately 3,600. Industry representation included e-commerce, SaaS software, financial services, media technology, and healthcare analytics. Cloud provider distribution included AWS-primary (nine organizations), GCP-primary (three organizations), and multi-cloud with no single majority (three organizations).

**Table 1. Participant distribution by annual cloud spend range, estimated waste percentage at study entry, and primary self-reported waste driver.**

| Cloud Spend Range | Orgs | Avg. Waste at Entry | Primary Waste Driver |
|---|---|---|---|
| $500K – $2M | 4 | 31% | Over-provisioned dev environments |
| $2M – $10M | 6 | 26% | Idle/orphaned resources + poor tagging |
| $10M – $30M | 3 | 24% | Kubernetes over-reservation |



ISSN: 2455-135X https://www.ijcsejournal.org/

| | | | |
|---|---|---|---|
| $30M+ | 2 | 19% | Reserved instance underutilization |

### *3.2 The Four-Pillar FinOps-DevOps Integration Model*

Analysis of participant approaches and available practitioner frameworks produced a four-pillar model for FinOps-DevOps integration. The pillars represent four distinct integration dimensions that together constitute a comprehensive FinOps-aware DevOps practice.

#### *3.2.1 Pillar 1 - Cost Visibility and Attribution*

The foundational pillar. No optimization is possible without knowing what is being spent, by whom, and on what. This pillar encompasses resource tagging and labeling strategies that attribute cloud costs to teams, services, and environments; integration of cost data into engineering dashboards alongside performance and reliability metrics; Kubernetes cost allocation for container workloads; and cost anomaly detection that surfaces unexpected spend increases in near-real-time rather than at month end.

The primary technical investment in this pillar is tag governance: defining a tagging taxonomy, enforcing it at provisioning time (through IaC linting, cloud provider tag policies, or admission control), and continuously auditing existing resources for compliance.

#### *3.2.2 Pillar 2 - Pipeline-Integrated Cost Awareness*

The integration of cost estimation and cost policy enforcement directly into CI/CD pipeline workflows. This pillar includes pull request cost difference estimates for IaC changes, cost policy gates that block or flag configurations exceeding defined thresholds, environment lifecycle automation, and deployment metadata that links cost allocation to specific deployment events.

This pillar shifts cost feedback from reactive to proactive. The cultural shift it requires engineering teams treating cost as a deployment quality signal alongside performance and security is often the harder part of the implementation.

#### *3.2.3 Pillar 3 - Continuous Resource Optimization*

The ongoing operational practice of identifying and acting on rightsizing opportunities, idle resource elimination, reserved capacity management, and Kubernetes efficiency improvement. This pillar includes automated utilization monitoring with rightsizing recommendations, idle resource detection and decommissioning workflows, spot and preemptible instance adoption for appropriate workloads, and scheduled scaling for predictable demand patterns.

#### *3.2.4 Pillar 4 - Financial Accountability Culture*

The organizational and cultural infrastructure that makes cost optimization self-sustaining rather than a periodic initiative. This pillar includes team-level cost budgets and variance alerts, show back and chargeback models that create financial accountability at the team level, unit economics tracking (cost per transaction, cost per user, cost per deployment), and regular cost review rituals.

### *3.3 Maturity Assessment*

Each organization was assessed across the four pillars at study entry and exit, with each pillar scored 0-3 (maximum composite score of 12). Entry composite scores ranged from 1 to 8, with a mean of 4.2. Exit scores ranged from 3 to 11, with a mean of 7.6. No organization reached full maturity across all four pillars by study exit, though two reached composite scores of 10 or higher.

### *3.4 Metrics Framework*

Six primary metrics were tracked throughout the study: cloud waste percentage, cost anomaly detection lead time, rightsizing adoption rate, tag coverage rate, non-production environment runtime fraction, and cloud cost per engineering output unit. These metrics were tracked quarterly using each organization's cost management tooling combined with utilization analysis.

## 4. Results and Analysis

### *4.1 Waste Reduction Outcomes by Pillar Maturity*

The 47% average cloud waste reduction cited in the abstract is the comparison between organizations that reached composite maturity scores of 8 or above by study midpoint and those that remained below 5 throughout. The absolute waste percentages declined across all maturity levels over the study period, but the differential between high and low maturity organizations widened.

**Table 2. Average cloud waste percentage at study entry and exit by composite maturity score range, with primary waste reduction mechanism.**

| Score Range | Waste % (Entry) | Waste % (Exit) | Primary Reduction Mechanism |
|---|---|---|---|
| 1–3 (Low) | 33% | 29% | Ad-hoc manual cleanup |
| 4–6 (Moderate) | 27% | 18% | Monitoring + selective rightsizing |
| 7–9 (High) | 23% | 11% | Automated detection + systematic rightsizing |
| 10–12 (Advanced) | 19% | 8% | Full pillar integration + team accountability |

The low-maturity organizations' modest improvement (33% to 29% waste) reflects the limitations of reactive management: periodic manual audits identify and remove obvious waste during the review period, but new waste accumulates between reviews. The advanced-maturity



ISSN: 2455-135X https://www.ijcsejournal.org/

organizations, with continuous automated monitoring and team-level accountability, showed a pattern of ongoing improvement across the full twenty months.

### *4.2 Cost Anomaly Detection: The Continuous Monitoring Advantage*

The comparison between automated continuous cost monitoring and periodic manual bill review on anomaly detection lead time was one of the most practically significant findings of the study. Organizations relying primarily on monthly bill review had a median anomaly detection lead time of 23 days. Organizations with automated continuous monitoring averaged a detection lead time of 4.2 hours.

**Table 3. Cost anomaly detection lead time, unexpected spend per anomaly, and estimated annualized savings by monitoring approach.**

| Monitoring Approach | Detection Lead Time | Unexpected Spend/Anomaly | Annualized Savings |
|---|---|---|---|
| Monthly manual review | 23.1 days | $41,200 | Baseline |
| Weekly automated reports | 6.4 days | $14,800 | $52,400 avg. |
| Daily automated alerts | 18.3 hours | $2,100 | $179,000 avg. |
| Continuous anomaly detection | 4.2 hours | $490 | $214,000 avg. |

### *4.3 The Rightsizing Execution Gap*

The finding most consistent across all fifteen organizations regardless of maturity tier was the gap between rightsizing recommendations generated and rightsizing recommendations implemented. Across the full study period, the average rightsizing adoption rate was 34%: for every three recommendations generated by cost optimization tooling, two were ignored or deferred indefinitely.

The reasons varied but clustered around several themes: risk aversion (the asymmetry of blame), ownership ambiguity (infrastructure vs. application teams), and process friction (change management overhead exceeding apparent value). The organizations with the highest rightsizing adoption rates had addressed all three barriers with specific mechanisms.

### *4.4 Kubernetes Cost Management*

Among the nine organizations running Kubernetes as their primary container platform, the median Kubernetes efficiency ratio actual CPU consumption as a proportion of total reserved CPU was 31% across these organizations, meaning 69% of reserved compute capacity was allocated but not actually consumed.

**Table 4. Kubernetes cost management practices, adoption rates, and average efficiency and cost outcomes.**

| Practice | Orgs | Efficiency Improvement | Monthly Savings/Cluster |
|---|---|---|---|
| OpenCost/Kubecost | 7/9 | N/A (visibility only) | N/A |
| VPA (recommendation mode) | 4/9 | +18 pts | $3,200 |
| VPA (auto update mode) | 3/9 | +27 pts | $5,800 |
| HPA tuning | 6/9 | +12 pts | $2,100 |
| Cluster Autoscaler + spot | 5/9 | N/A (node cost) | $7,400 |
| Namespace budget limits | 4/9 | Prevention-focused | $1,900 |

### *4.5 Environment Lifecycle Automation*

Non-production environment runtime fraction at study entry averaged 84% across all fifteen organizations. Simple back-of-envelope math illustrates the waste: if non-production environments could be shut down during the 76% of hours when they are unused, and if those environments represent a typical 25-30% of total cloud spend, the potential savings are significant.

The average non-production environment runtime fraction for organizations that implemented lifecycle automation dropped from 84% to 38% over the study period. For the organizations in the study, this translated to average non-production environment cost reduction of 44%.

### *4.6 The Tagging Tax: Why Cost Attribution Fails*

Tag coverage rate at study entry across all fifteen organizations averaged 61%, meaning 39% of cloud resources were carrying incomplete or absent tagging. Organizations that had implemented both IaC-level enforcement and periodic drift correction by study exit averaged a tag coverage rate of 91%, up from 58% at entry. Organizations with enforcement only averaged 73%.

## 5. Conclusion

Cloud costs are an engineering problem as much as a financial one, and the organizations that treat them as such bringing cost visibility into engineering workflows, automating cost policy enforcement, and building team-level financial accountability consistently outperform those that treat cloud spending as a finance team responsibility reviewed periodically. The 47% average cloud waste reduction among high-maturity organizations in this study is not a theoretical ceiling. It is an observed outcome from practical implementation across organizations of varying sizes and sectors.



ISSN: 2455-135X https://www.ijcsejournal.org/

The four-pillar model proposed here provides a practical framework for organizations building FinOps-DevOps integration. Cost Visibility and Attribution (Pillar 1) must precede everything else. Pipeline-Integrated Cost Awareness (Pillar 2) converts cost data from historical reporting into proactive decision support. Continuous Resource Optimization (Pillar 3) executes on the opportunities Pillar 1 identifies. Financial Accountability Culture (Pillar 4) sustains the practice when organizational attention shifts.

Continuous automated anomaly detection is the single highest-leverage intervention available to organizations that have not yet implemented it. The detection lead time differential between monthly manual review and continuous monitoring 23 days versus 4.2 hours directly translates to unexpected spend reduction that typically covers the cost of the monitoring platform many times over.

The rightsizing execution gap the persistent 66% of recommendations that go unimplemented is the industry's most visible evidence that FinOps tooling is necessary but not sufficient. Organizations that build mechanisms to address risk aversion, resolve ownership ambiguity, and reduce process friction achieve rightsizing adoption rates three to four times higher than those relying on recommendation generation alone.

Future research should examine the long-term sustainability of FinOps programs beyond two-year horizons, and the relationship between FinOps maturity and engineering productivity.

## 6. Conflicts of Interest

The author(s) declare(s) that there is no conflict of interest regarding the publication of this paper.

## Funding Statement

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

## Acknowledgments

The authors wish to acknowledge the contributions of the fifteen participating organizations and the FinOps Foundation community.

## References

[1] FinOps Foundation, "FinOps Framework: Phases, Capabilities, and Personas," FinOps Foundation, 2024. [Online]. Available: https://www.finops.org/framework/

[2] Flexera, "2024 State of the Cloud Report," Flexera Software LLC, 2024.

[3] M. Castellanos and M. Harmon, "From Visibility to Value: Organizational Patterns in Cloud Cost Management Maturity," Journal of Cloud Computing Research, vol. 14, no. 2, pp. 88–103, 2022.

[4] R. Kelly, "The FinOps Handoff Problem: Organizational Barriers to Cloud Cost Optimization in Large Technology Firms," Harvard Business School Case Study HBS-923-014, 2023.

[5] McKinsey & Company, "Capturing the Value of Cloud Cost Optimization," McKinsey Digital, 2022.

[6] Kubecost/CNCF OpenCost, "OpenCost: The Open Source Standard for Kubernetes Cost Monitoring," 2024. [Online]. Available: https://www.opencost.io/

[7] Infracost, "The Business Case for Pull Request Cost Visibility: Analysis of Cost Incident Reduction," Infracost Research Report, 2023.

[8] Amazon Web Services, "AWS Cost Anomaly Detection: Reducing Cloud Cost Surprises," AWS Documentation, 2024. [Online]. Available: https://docs.aws.amazon.com/cost-management/

[9] Google Cloud, "Cloud Billing: Cost Management and Optimization Best Practices," Google Cloud Documentation, 2024. [Online]. Available: https://cloud.google.com/billing/docs/

[10] Kubernetes SIG Autoscaling, "Vertical Pod Autoscaler: Automated Resource Recommendations for Kubernetes Workloads," 2024. [Online]. Available: https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler

[11] G. Kim, J. Humble, P. Debois, and J. Willis, The DevOps Handbook. Portland, OR: IT Revolution Press, 2016.

[12] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA: O'Reilly Media, 2016.

[13] J. Storment and M. Fuller, Cloud FinOps: Collaborative, Real-Time Cloud Financial Management. Sebastopol, CA: O'Reilly Media, 2019.

[14] J. Shafer, "Engineering Economics for Cloud Infrastructure: Unit Cost Modeling in DevOps Environments," IEEE Software, vol. 40, no. 4, pp. 32–41, 2023.

[15] CNCF, "FinOps for Kubernetes: Operational Guidance for Cloud Cost Efficiency," Cloud Native Computing Foundation Technical Advisory Group Report, 2023.

[16] HashiCorp, "Terraform Cloud Cost Estimation: Surfacing Infrastructure Cost in Plan Output," 2024. [Online]. Available: https://developer.hashicorp.com/terraform/cloud-docs/cost-estimation

[17] T. Gilmore and P. Walsh, "Spot Instance Adoption Patterns and Risk Management in Production Kubernetes Environments," SREcon Americas 2022, 2022.

[18] C. Dolan, "Resource Tagging Governance at Scale: Enforcement Strategies for Multi-Account Cloud Environments," DevOps Enterprise Summit 2024, 2024.

[19] N. Forsgren, J. Humble, and G. Kim, Accelerate: The Science of Lean Software and DevOps. Portland, OR: IT Revolution Press, 2018.

[20] ProsperOps, "Annual State of Cloud Commitments: Reserved Instance and Savings Plan Optimization Benchmark Report," ProsperOps Research, 2024.