O'REILLY®

Second Edition

Cloud FinOps

Collaborative, Real-Time Cloud Value Decision Making

Compliments of

vmware®

J.R. Storment & Mike Fuller

O'REILLY®

Cloud FinOps

FinOps brings financial accountability to the variable spend model of cloud. Used by the majority of global enterprises, this management practice has grown from a fringe activity to the de facto discipline managing cloud spend. In this book, authors J.R. Storment and Mike Fuller outline the process of building a culture of cloud FinOps by drawing on real-world successes and failures of large-scale cloud spenders.

Engineering and finance teams, executives, and FinOps practitioners alike will learn how to build an efficient and effective FinOps machine for data-driven cloud value decision-making. Complete with a road map to get you started, this revised second edition includes new chapters that cover forecasting, sustainability, and connectivity to other frameworks.

You'll learn:

- A road map to build a culture of FinOps with executive support
- How to understand and forecast your cloud spending
- How to empower engineering and finance to partner
- Cost allocation strategies to create accountability for cloud and container spend
- Strategies for rate discounts from cloud commitments
- When and how to implement automation of repetitive cost tasks
- How to empower engineering team action on cost efficiency
- How to use unit economics to guide data-driven decision-making

"Adoption of FinOps requires both executive and ground level support. This edition of the definitive guide to FinOps lays out playbooks for how to create a partnership with engineering teams, how to accurately forecast spend, using data from other frameworks to influence decisions, and how ultimately cultural changes trump mandates."

—**Beth Marki**
Executive Director of Cloud Financial Management at JPMorgan Chase

J.R. Storment is Executive Director of the FinOps Foundation. He works with the largest cloud consumers in the world to advance their people through community, education, and standards.

Mike Fuller is principal engineer on the cloud FinOps team at Atlassian in Australia, developing FinOps practice and enabling engineering teams to efficiently remain in control of cloud spend.

CLOUD COMPUTING

Twitter: @oreillymedia
linkedin.com/company/oreilly-media
youtube.com/oreillymedia

ISBN: 978-1-098-13532-4

vmware

Explore All the Ways Your Organization Can Benefit from Cloud FinOps Best Practices with VMware

Make sense of your cloud data

Optimize and control your cloud spend

Enhance your cloud management practice

VMware Aria Cost™ powered by CloudHealth has helped practitioners from more than 20,000 organizations around the world build, manage and optimize their FinOps practice.

FinOps FOUNDATION Premier Partner

Learn more at **cloudhealth.vmware.com**

Praise for *Cloud FinOps*

Security is Job 0, FinOps is Job 0.5. Just as security is everyone's job, so too is FinOps. The second edition of the *Cloud FinOps* book provides a recipe for FinOps success.

—Marit Hughes, Deloitte

FinOps brings business, finance, and engineering together to drive a culture of transparency and accountability for cloud spend management. Google Cloud and the Finops Foundation are working together to pioneer open billing standards for further transparency and acceleration of cloud business value. This book synthesizes those collaborative efforts.

—Mich Razon, VP & GM, Head of Commerce Platforms at Google Cloud

Optimizing cost is critical to the success of cloud migrations and the move to consumptive models. Cloud optimization should be a key area of focus for companies. The FinOps book lays out a road map for how to drive the cultural change that is required across the technology, finance, and business organizations of an enterprise running at scale in the cloud.

—Fred Delombaerde, VP Core Commerce at Microsoft

Today, public cloud cost management transcends any individual vendor. Large organizations need to normalize and merge a variety of vendor costs and usage data into a single view, to enable data-driven decisions by engineering, business, and finance. It is essential that the FinOps Foundation continues to iterate on the FinOps framework and facilitate the creation of an open standard that vendors can then use to consistently present their cost and usage data, to enable the work of FinOps practitioners.

—Udam Dewaraja, Head of Cloud Financial Management at Citi

Any company looking to optimize cloud costs should first turn to the *Cloud FinOps* book and second to the materials and community of the FinOps Foundation. Together these resources have helped to lay a successful foundation for Target Corporation's FinOps practice. We have leveraged the FinOps domains and capabilities to build tools for leaders and engineers. With the seven new chapters included in the Cloud FinOps second edition, including key concepts like putting "Data in the Path of the Engineer" and leveraging the "UI of FinOps," we will mature our FinOps practice and adoption for our public and private cloud utilization.

—Kim Wier, Director of Engineering, Target Corporation

FinOps as a theory may be hard for some execs to grasp. Leadership is much more likely to back an effort that has already shown demonstrated, quantified success. Prove the value and you'll get the backing you seek. The new chapters in the FinOps book, such as the "UI of FinOps" and "Adopting FinOps," give invaluable insights into how to drive engineering action on cost efficiency and gain executive support.

—Jason Rhoades, Intuit

Adoption of FinOps requires both executive and ground level support. This edition of the definitive guide to FinOps lays out playbooks for how to create a partnership with engineering teams, how to accurately forecast spend, using data from other frameworks to influence decisions, and how ultimately cultural changes trump mandates.

—Beth Marki, Executive Director of Cloud Financial Management at JPMorgan Chase

FinOps has emerged as a new model to define the future of how finance and technical teams partner together. The *Cloud FinOps* book has provided a road map for those organizations looking to evolve the way they manage and optimize cloud expenditures, without slowing technical teams and innovation. This is a must-read for both finance and technical teams to help them understand their role in the world of cloud financial management!

—Keith Jarrett, Cloud Financial Management Leader

SECOND EDITION

Cloud FinOps

Collaborative, Real-Time Cloud Value Decision Making

J.R. Storment and Mike Fuller

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Cloud FinOps, Second Edition
by J.R. Storment and Mike Fuller

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: John Devins
Development Editor: Corbin Collins
Production Editor: Kate Galloway
Copyeditor: nSight, Inc.
Proofreader: Piper Editorial Consulting, LLC

Indexer: nSight, Inc.
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Kate Dullea

December 2019: First Edition
January 2023: Second Edition

Revision History for the Second Edition
2023-01-18: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781492098355* for release details.

978-1-098-13532-4

[LSI]

For Our Families

To Jessica and Oliver of the Triangle for being patient and supportive the second time around too. And to Wiley, whose impact we feel daily.
—J.R.

Lesley, Harrison, and Claire, without your love and support this would have never been possible.
—Mike

Table of Contents

Part II. Inform Phase

Part III. Optimize Phase

Preface

Dear Reader,

J.R. and Mike here, the coauthors of the book you are holding. We began this second edition undertaking in early 2022. When we wrote the first edition of this book in 2019, the world was a different place. COVID had massively accelerated the move to the cloud, shattering all previous estimates of cloud spend growth. After we started writing this edition, a global economic downturn forced organizations—including the major cloud service providers—to elevate the importance of cloud spend efficiency to board-level discussions. Amazon's CFO has pledged to help customers trim their cloud bills, and Microsoft's CEO indicated that their top priority is to help their customers do more with less.

The World Turned Upside Down

FinOps as a concept went from an obscure term to a business necessity. At conferences pre-2020 we frequently overheard: "What the heck is FinOps?" By 2023 it is predicted that 80% of organizations using cloud services will establish a dedicated FinOps function. The FinOps Foundation went from a twinkle in our eyes to a global force with nearly 10,000 practitioners of the craft representing every major industry and the majority of the Fortune 500, FTSE 100, and ASX 250. FinOps went from being a fringe term to receiving 10 times the search traffic and analyst mentions that the "cloud financial management" moniker pushed by large clouds like Amazon Web Services did.

The pay-as-you-go model—also known as the *variable spend model*—allows engineers to gain rapid access to infrastructure so that they can innovate quickly. The stories we hear remain: engineering teams still consume resources in the cloud with not enough prioritization of cost efficiency. Finance teams struggle to understand and keep up with what teams are spending and to properly allocate cloud investments. Leadership still doesn't know which levers to pull to implement a FinOps transformation and drive more guidance of cloud spend investment.

The problem has only gotten more complex in the last three years. Cloud providers have added hundreds of thousands of additional product stock-keeping units (SKUs) driven by thousands of new features. In the months preceding publication of this edition, Google Cloud and Azure offered completely new ways of making discount commitments in the form of Azure Savings Plans and Google Flexible Committed Use Discounts (Flexible CUDs), changing the game yet again.

During the last decade of our respective careers, we heard a consistent theme from practitioners and executives alike: there's a lack of cloud value education and knowledge available. Mike heard it while running cloud cost optimization for Atlassian's massive cloud deployments. J.R. heard it while coaching the world's largest cloud spenders as cofounder of Cloudability's cloud spend management platform (acquired by Apptio in 2019) from 2011 to 2020 and later as the executive director of the FinOps Foundation, as an employee of the Linux Foundation.

From this demand for a resource from which we can all learn, along with a need for someone to formally define FinOps, we joined forces in early 2019 with 26 expert practitioners from the industry to create the FinOps Foundation.

The community of now nearly 10,000 practitioners from the FinOps Foundation have provided most of the best practices we'll cover in this book. The examples herein come from their hard-earned experience and war stories shared in the various community forums. We've also woven in quotes from them to help connect the content with real-life thoughts and opinions on FinOps.

Who Should Read This Book

Anyone working in engineering, finance, procurement, product ownership, or leadership in a company running—or aspiring to run—in the public cloud will benefit from this book. As an organization understands the personas in FinOps, it can map them to relevant teams across the business.

Engineers are most likely not used to thinking about costs as a day-to-day concern. In the precloud days, they were mainly worried about performance in relation to hardware. Constrained by procurement and unable to get more servers whenever they needed them, they had to plan far in advance. They have to think about the cost of their infrastructure choices and its impact on the business. At first, this can feel foreign and at odds with their primary focus of shipping product and feature enhancements. Over time they add cost as another efficiency metric that they can tune to positively impact the business.

> *A lot of engineers will just throw hardware at a problem. FinOps requires engineers to consider the cost and margins.*
>
> —John Stuart, VP, DevOps, Security & IT at Jobvite

Finance traditionally focused on retroactive reporting on a monthly or quarterly granularity, based on set budgets that quickly became out of date. The job has evolved now to help enable the business to continually move forward, and to proactively partner with tech and engineering teams to forecast spend, based on the actions of engineers (who aren't used to thinking about that cost). In other words, they're shifting from opaque and fixed capital expenditure (CapEx) reporting to transparent and fluid operational expenditure (OpEx) forecasting. As part of that role, finance teams become engineering team partners who understand what the drivers of cloud spend are amid thousands of SKUs. They help to fundamentally reinvent how to perform the finance function, while also rethinking how to report technology spend to executives and investors.

Procurement teams are used to tightly controlled spending, careful rate negotiations, and wielding the power of the purchase order before vendors get paid. Now procurement teams become strategic sourcing. They pull all rogue spending together into an enterprise agreement with their cloud service provider to get the best rates for what engineers are already using to deliver value.

> *We don't win by shaving cents from the cloud provider. We win by delivering features to our customers.*
>
> —Alex Landis, Autodesk

Product owners, responsible for the pricing and margins of their products, struggled in the precloud world to understand all the costs that went into servicing a customer or operating a commercial offering. Cloud lets them understand more, or all, of the cost of delivering digital value, understand the cost impact of new features, and see how customer use cases create variability in the cost of an application. This allows product teams to collaborate much more effectively with engineering teams developing the products, to forecast revenue and margin more effectively, and to target customer demand much more efficiently.

Leadership, C-level executives, VPs, directors, or technology leaders managing budgets for a team have often lost direct control of cloud spending decisions and now rely on their teams to operate within reasonable budgets. Tech executives no longer plan large purchase decisions far in advance. Instead, they think more about how to forecast and manage spending that's already happening. The conversation has shifted from ensuring services have the capacity to ensuring that they're spending the right amount of money for the job. They want more control over how much is spent and more ability to strategically influence where it is spent.

This book seeks to break down barriers between these personas by laying out a common lexicon and set of best practices to follow.

About This Book

In the coming chapters, we'll formally define FinOps. The definition we've created was formulated by the most experienced cloud financial management teams on earth, who manage hundreds of millions of dollars (or billions in some cases) per year in cloud spend. We've collected their common practices for achieving cloud success along with some pitfalls they've identified and solved. We'll show what effective FinOps looks like and how to start the FinOps transformation in your organization.

Previously, the only way to gain access to this knowledge would be to attend public events where these experts would present their ideas. This book and the FinOps Foundation changes that with a vibrant community, in-depth training resources, and standardized best practices.

After you've read this book, we encourage you to get involved with the FinOps Foundation's various programs as a pathway to continue honing your skills and career. The mission of the foundation is to advance every individual who manages the value of the cloud. The community is there for you. We hope that the real-world strategies, processes, and stories in this book will inspire everyone to better take control of cloud spend. And in the process, we can make our organizations, and our own careers, more competitive.

What You Need to Know Before Reading On

At the time of writing, we assume readers will have a base level of knowledge of at least one of the three main public cloud providers (Amazon Web Services [AWS], Azure, and Google Cloud Platform [GCP]). Readers should understand how the cloud works and charges for resources. They should also be familiar with the major resource types like compute and storage, and higher-level service offerings like managed databases, queues, and object storage.

A good starting place for the level of AWS knowledge needed is the AWS Business Professional training, or better, the AWS Cloud Practitioner certification. Both cover the basics of operating in AWS. For Google, check out the GCP Cloud Digital Leader course. For Azure, try the Azure Fundamentals learning path. These can usually be completed in a single-day workshop or through online training.

Readers should also understand the fundamentals of how cloud computing works; know the key services on their cloud provider, including their common use cases; and have a basic understanding of how billing and pricing work in the pay-as-you-go consumption model.

For example: as an AWS user, you should already know the difference between EC2 (Elastic Compute Cloud) and RDS (Relational Database Service). You should understand that there are different ways to pay for those resources, such as On-Demand,

Reserved Instances (RIs), Savings Plans (SPs), and Spot. It's OK if you don't know how RIs or SPs work in detail or how to plan a strategy for purchasing them—we're going to cover that—but you should already understand that they can be used to save money on compute resources.

FinOps Is Evolving

Over the past few years, so much has evolved into what we call FinOps today—and it will continue to evolve. As the cloud service providers offer more and more services and continue to offer different ways to optimize their platforms, FinOps will keep adapting. We recommend always confirming the details of the cloud service provider offerings we explore throughout this book. If there are corrections, alternative opinions, or criticisms of anything in this book, we encourage readers to contact us. After all, it has taken brave folks challenging the way things are done to help formulate the successful FinOps practices we have today.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

O'Reilly Online Learning

For more than 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *https://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/cloud-finops-2e*.

Email *bookquestions@oreilly.com* to comment or ask technical questions about this book.

For news and information about our books and courses, visit *https://oreilly.com*.

Find us on LinkedIn: *https://linkedin.com/company/oreilly-media*.

Follow us on Twitter: *https://twitter.com/oreillymedia*.

Watch us on YouTube: *https://youtube.com/oreillymedia*.

Acknowledgments

First, to our families (Lesley, Jessica, Harrison, Oliver, and Claire), who sacrificed so many nights and weekends in allowing us to write this book originally and then again for the second edition, and who suffered through us talking about nothing else but FinOps: thank you.

To the team at O'Reilly (Corbin Collins, John Devins, Kate Galloway, Kristen Brown, and Sara Hunter): without your efforts this never would have happened.

Rob Martin, who has reread, commented, and proposed more thoughtful (and often verbose!) changes than we could even consider.

Thanks to the teams at Amazon Web Services, Google Cloud, and Microsoft Azure for your ongoing feedback and reviews.

Tech reviewers: Jason Rhoades and Joshua Bauman, who stepped in quickly to power through a hornet's nest of comments, revisions, and suggestions on our nearly final draft.

We would like to thank all of the members of the FinOps Foundation. Since starting the FinOps Foundation, we've been humbled by the number of companies and practitioners signing up and by the willingness of charter members to assist in formalizing what FinOps is.

To the FinOps Foundation staff: Andrew Nhem, Ashley Hromatko, Ben de Mora, Joe Daly, Kevin Emamy, Kyle McLaughlin, Natalie Bergman, Rob Martin, Ruben Vander Stockt, Samantha White, Stacy Case, Steven Melton, Steven Trask, Suha Shim, Thomas Sharpe, and Vasilio Markanastasakis. Your efforts to help build the foundation are greatly appreciated.

Mike would like to thank his FinOps team at Atlassian. Daniel Farrugia, Diana Mileva, Florence Timso, Letian Wang, Sara Gadallah, Teresa Meade, and Tom Cutajar.

Throughout the years we've worked with many people who have shared their companies' stories around their cloud challenges and the methods they have implemented to solve them. There are too many to mention, but to all of you: thanks.

Finally, thank you to all the people who helped along the way; without your efforts this book would not be what it is today: Aaron Edell, Abuna Demoz, Adam Heher, Alex Hullah, Alex Kadavil, Alex Landis, Alex Sung, Alexander Price, Alison McIntyre, Alison Pumma, Ally Anderson, Amelia Blevins, Anders Hagman, Andrew Midgley, Andrew Thornberry, Anthony Tambasco, Antoine Lagier, Ben Kwan, Benjamin Coles, Bhupendra Hirani, Bindu Sharma, Bob Nemeth, Brad Payne, Casey Doran, Dana Martin, Darek Gajewski, David Andrews, David Angot, David Arnold, David Shurtliff, David Sterz, David Vale, Dean Layton-James, Dieter Matzion, Elliot Borst, Elliott Spira, Ephraim Baron, Erik Onnen, Gavin Cahill, Geoffrey Anderson, Ilja

Summala, James Jackson, Jason Fuller, Jerome Hess, Jess Belliveau, John McLoughlin, John Merritt, John Stuart, Jon Collins, Justin Kean, Keith Jarrett, Ken Boynton, Lindbergh Matillano, Manish Dalwadi, Mark Butcher, Mark Richter, Marsha Shoemaker, Martin Bleakley, Mat Ellis, Matt Finlayson, Matt Leonard, Michael Flanakin, Michele Allesandrini, Nabil Zakaria, Naveen Chand, Pedro Silva, Phillip Coletti, Renaud Brosse, Rich Hoyer, Rick Ochs, Sarah Grey, Sascha Curth, Shane Anderson, Stephanie Gooch, Stephen Elliot, Tom Cross, Tom March, Tom Marrs, Tony Curry, Umang Sehgal, Virginia Wilson, Webb Brown, Wendy Smith, and William Bryant.

Cheers to everyone who allowed us to quote them throughout the book.

PART I

Introducing FinOps

In the first part of this book, we cover a lot of the fundamentals, such as what FinOps is, how teams operate, the language of FinOps, how cloud services are billed, and a roadmap to start your own FinOps journey guided by the FinOps Foundation Framework.

CHAPTER 1

What Is FinOps?

In the simplest terms, FinOps brings financial accountability to the variable spend model of cloud. But that description merely hints at the outcome. The cultural change of running in the cloud moves ownership of technology and financial decision making out to the edges of the organization, from procurement to engineering, architecture, and product teams. It empowers technology, finance, and business professionals to work together in more efficient ways.

Organizations around the world are moving aggressively to cloud, whether they are ready or not. Like the transition from mainframes to client/server, to the web, or to mobile, the shift to cloud (and from projects to products) brings with it massive changes to the way traditional business functions need to account for, request, and manage technology costs.

FinOps demands you acknowledge that the old ways of managing infrastructure aren't just ineffective; they create situations where runaway cloud costs can threaten the business. While traditional infrastructure management methodologies will still be needed for owned infrastructure in the future, no level of adherence to these methodologies—which don't handle the variable consumption model of cloud well—will allow organizations to manage cloud effectively without the introduction of FinOps.

Defining the Term "FinOps"

As of early 2023, the FinOps Foundation defines it as such:

> FinOps is an evolving cloud financial management discipline and cultural practice that enables organizations to get maximum business value by helping engineering, finance, technology, and business teams to collaborate on data-driven spending decisions.

At its core, FinOps is a cultural practice. It's the way for teams to manage their cloud costs, where everyone takes ownership of their cloud usage supported by a central best-practices group. Cross-functional teams in engineering, finance, product, procurement, etc. work together to enable faster product delivery, while at the same time gaining more financial control and predictability.

FinOps is a portmanteau of the words "Finance" and "DevOps," stressing the communications and collaboration of business and engineering teams. Cloud FinOps is sometimes incorrectly referred to as "Cloud Financial Operations," but that term is falling out of favor due to its ambiguity with the more traditional "Financial Operations" role that exists in finance. Other synonyms for the practice include "Cloud Financial Management," "Cloud Financial Engineering," "Cloud Cost Management," or "Cloud Optimization." Whatever you call it, FinOps is the practice of getting the most business value out of your cloud spend.

This chapter discusses the core principles of FinOps, how the associated cultural transformations began, and why every organization needs to embrace the discipline for cloud success.

But first, let's set the stage for defining FinOps with a typical story of an individual practitioner's journey.

The FinOps Hero's Journey

Today's FinOps leader often comes out of a world of managing, planning, and accounting for traditional IT and virtualized servers. Here's a typical story that is an amalgamation of those we've heard over the years. It's likely you are about to embark on a similar journey, are currently on it, or have completed parts of it already. The hero in this story is called Finn.

Things were pretty straightforward for Finn: backward-looking financial reports were done quarterly, and capacity planning meant extrapolating usage trends to guess the production needs of the organization for the next few quarters to meet changing demands for its products. There weren't a lot of surprises in spending.

Then Finn noticed an increasing number of AWS or Google Cloud payables coming in without purchase orders attached. One of his cloud-savvy colleagues, Ana, explained the highly variable nature of cloud and how it's just, well, different than on-premises data centers—and that there's an entirely new way of managing it, as well as a new professional discipline emerging to do so.

Finn carefully considered Ana's words. It did sound interesting, and appealing. But then Finn remembered how well the processes he had in place for the organization's 8,000

on-premises servers work. Cloud couldn't be that different. Surely if he just applied his existing processes more diligently, cloud costs could be managed as well.

The next quarter, cloud spending doubled unexpectedly, and Finn went back to Ana with his tail between his legs. He committed to trying a new set of processes that look more frequently at spend and increasing his interface time with the engineering teams creating the spend.

All of a sudden the finance leader, who previously never cared about cloud spending, began pushing Finn to go back to quarterly reporting, saying the real-time approach he was taking didn't fit with the company's other processes. The technology leader was pushing back, saying she couldn't consider cost and also make her product delivery deadlines. Finn's executive team encouraged a top-down control methodology. Finn again went back to Ana for help, and she led him to fellow journeyers at the FinOps Foundation. Learning from their mistakes and wins, Finn began to lay out a plan for how to reset the company's processes and, more ambitiously, effect cultural change.

It was go time. Finn rolled out a new cloud spend allocation strategy, tagging guidelines, criteria for rightsizing (i.e., resizing cloud resources to better match workload requirements), and an initial rate optimization commitment to his cloud provider. There was finally a path forward that seemed to allow him to account for the spend and the teams to get the tech they needed without friction. Cloud migration began to soar.

Right when things seemed to be going well, a new CFO came in and said the cloud was too expensive at scale, advocating for a widespread return to the data center. It was time for Finn's biggest test: working with his cloud-savvy colleagues to show that cloud is more than a cost center. They had to show how cloud can enable innovation and velocity in ways that on-premises data centers cannot, driving competitive advantage for the company. The CEO saw the bigger picture and agreed, paving the way for a cloud-first strategy.

Newly confident, Finn forged ahead, breaking down silos between teams and helping to drive real change in the organization. But he faced one last battle: cloud spend was now hitting material levels, affecting the bottom line. The CFO stepped in to stop the upward trend by any means necessary to ensure the organization's margins were not affected. Finn moved beyond the one-dimensional view of looking only at cloud spend and shifted to a unit economics model that tied the spend back to business value, giving him the context to clearly demonstrate that cloud spend was on the right path.

In the end, Finn realized that this journey is just the beginning. Cloud is constantly evolving, and FinOps with it. Finn and Ana will have the most influence in this new field by helping to define its best practices, something they see as a key way to give back to the community that helped them on their own journeys.

Some aspects of this journey may resonate with you, and others may be part of your journey ahead. Now that you've heard a typical story, let's look at where FinOps emerged.

Where Did FinOps Come From?

Trailblazers like Adobe and Intuit, early scalers in public cloud, provided the first glimpse to the coauthors of what would become FinOps as far back as 2012 in the Bay Area. In the mid-2010s, they interacted with larger companies like GE and Nike as they began their own early FinOps journey tied to their rapid expansion of cloud usage. A couple of years later, Mike saw forward-looking enterprises in Australia, like his own Atlassian as well as others like Qantas and Tabcorp, begin similar practices. Finally, during J.R.'s two-year tour of duty in London from 2017 to 2019, he was a firsthand witness to enterprises like BP, HSBC, and Sainsbury's as they developed this new approach across their company cultures. FinOps came into being slowly and simultaneously, all over the world, as the financial and accountability challenges of cloud presented themselves at scale everywhere, as various industries and regions began to adopt material amounts of cloud. Figure 1-1 traces the increasing prevalence of the term.

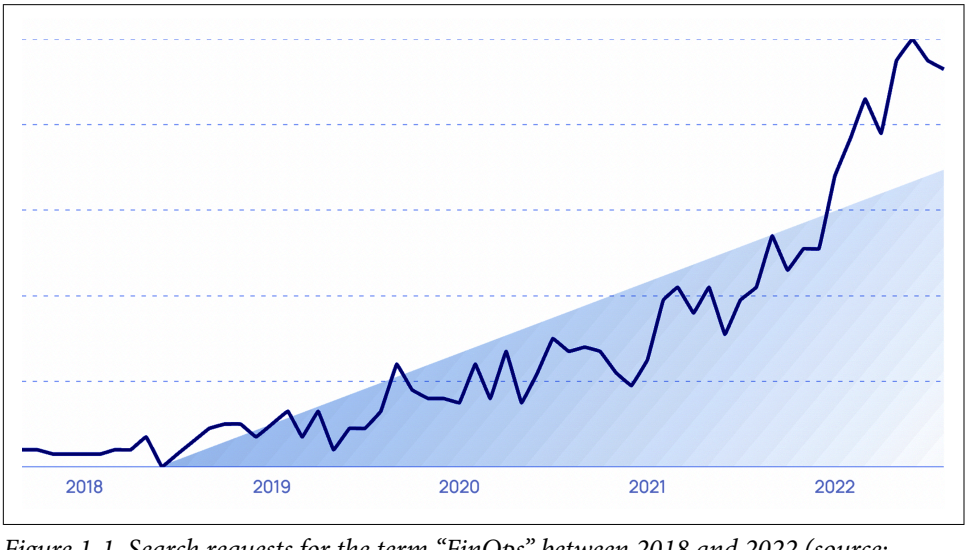


Figure 1-1. Search requests for the term "FinOps" between 2018 and 2022 (source: speakeasystrategies.com)

"FinOps" is a term that has come late to the party. In the early days, companies simply called the practice "cloud cost management." Later, "cloud cost optimization" began to take hold, although it didn't speak to the allocation challenges of cloud. AWS and other cloud providers began using the phrase "cloud financial management," a

catch-all title that is increasingly being replaced by "FinOps" based on search and analyst mentions referenced in the figures. Others, like the retailer Target, call their internal practice "Engineering Efficiency." Figure 1-2 shows increasing growth in "FinOps" mentions by industry analysts.

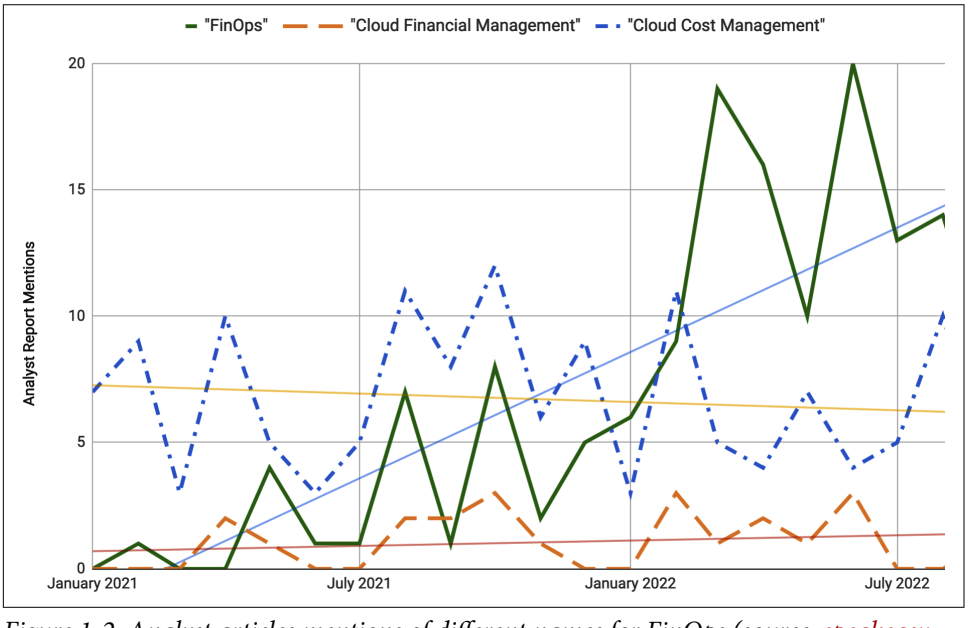


Figure 1-2. Analyst articles mentions of different names for FinOps (source: speakeasy-strategies.com)

In Figure 1-2, you can see the recent increase in cloud analyst articles using the term "FinOps" to describe this field of practice, as older terms like "cloud financial management" reduce in mindshare. Choosing this compound term, which purposely echoes DevOps, brings the vital cross-functional and agile aspect of the movement to the forefront.

And now FinOps is becoming a standalone profession worldwide. Figure 1-3 shows the growth in FinOps being a skill listed by members of the LinkedIn platform. The 2022 State of FinOps data showed that nearly every major industry, from Financial Services to Retail to Manufacturing and beyond, is now running a FinOps practice.

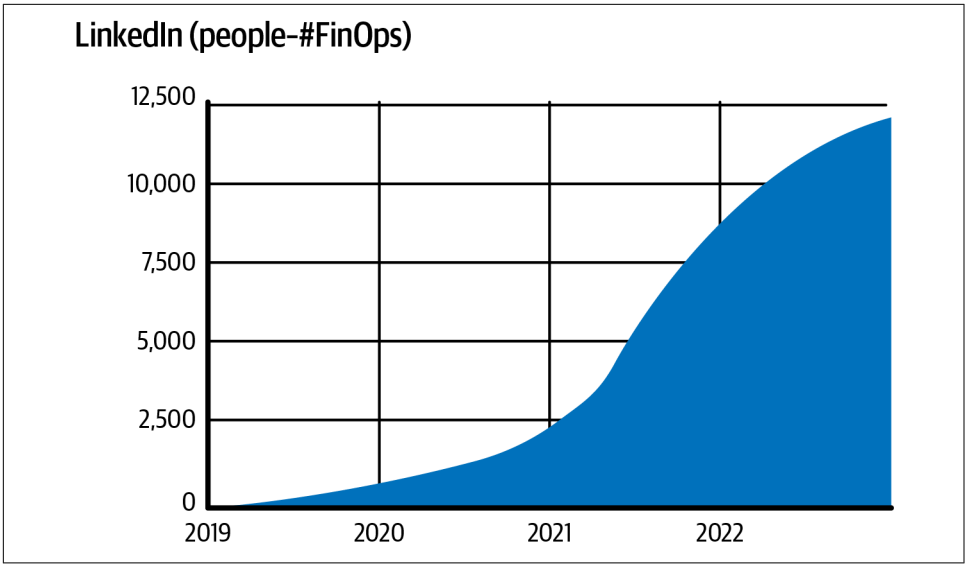


Figure 1-3. Count of people on LinkedIn listing FinOps as a skill set

State of FinOps data in 2021 indicated that 95% of those working in the field saw it as their likely career path.

Public job listings for FinOps practitioners at Fortune 500 enterprises ranging from Apple to Disney to CVS Health to Nike are popping up frequently on LinkedIn and the FinOps Foundation site.

The challenge today is less of finding a job in FinOps and more of companies finding enough skilled candidates to fill the roles they have, driving an explosion of those seeking FinOps certification and related skills.

Stories from the Cloud—J.R.

I first spoke about the concept of FinOps in a DevSecOps talk[1] with Emil Lerch from AWS at the AWS Public Sector Summit in Washington, DC, back in 2016. We started with the definition of DevOps from the venerable Gene Kim, author of *The Phoenix Project*:

> The term "DevOps" typically refers to the emerging professional movement that advocates a collaborative working relationship between development and IT operations, resulting in the fast flow of planned work (i.e., high deploy rates) while simultaneously increasing the reliability, stability, resilience, and security of the production environment.

Then, with a bit of hubris, or as an homage, I crafted a definition of FinOps based on Kim's:

> The term "FinOps" typically refers to the emerging professional movement that advocates a collaborative working relationship between *DevOps and finance*, resulting in *an iterative, data-driven management of infrastructure spending (i.e., lowering the unit economics of cloud)* while simultaneously increasing *the cost efficiency and, ultimately, the profitability of the cloud environment*.

Since then, the definition of FinOps has evolved and broadened, but it has retained the all-important principles of driving a collaborative working relationship between teams, making iterative changes using data-driven insights, and improving unit economics.

Data-Driven Decision Making

With FinOps, each operational team (workload, service, product owner) can access the near-real-time data they need to influence their spend and help them make data-driven decisions that result in efficient cloud costs balanced against the speed/performance and quality/availability of services.

> *If you can't out-experiment and beat your competitors in time to market and agility, you are sunk.... So the faster you can get those features to market and test them, the better off you'll be. Incidentally, you also pay back the business faster for the use of capital, which means the business starts making money faster, too.*
>
> —Gene Kim, *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win* (IT Revolution Press, 2013)

1 Emil Lerch and J.R. Storment, "Leveraging Cloud Transformation to Build a DevOps Culture," AWS Public Sector Summit, June 20, 2016, *https://oreil.ly/DCqfg*.

If it seems like FinOps is about *saving* money, then think again. FinOps is about *making* money. Cloud spend can drive more revenue, signal customer base growth, enable more product and feature release velocity, or even help shut down a data center.

FinOps is all about removing blockers; empowering engineering teams to deliver better features, apps, and migrations faster; and enabling a cross-functional conversation about where to invest and when. Sometimes a business will decide to tighten the belt; sometimes it'll decide to invest more. But now teams know why they're making those decisions.

Real-Time Feedback (aka the "Prius Effect")

There are three parts to a successful FinOps practice:

> Real–time reporting + just–in–time processes + teams working together = FinOps

We'll get into the second two later in the book. Right now, let's look at the first part.

The feedback loop of real-time reporting is a powerful influence on human behavior. In our experience, you should provide engineers with feedback on the impacts of their actions as close as possible to the time those actions occur. This tends to create automatic behavioral changes for the better.

Anyone who's driven an electric car has probably experienced the *Prius Effect*. When you put your foot down heavily on the pedal, an electric car's display shows energy flowing out of the battery into the engine. When you lift your foot up, energy flows back into the battery. The feedback loop is obvious and instantaneous. You can see how the choice you're making in the moment—one that in the past may have been unconscious—is impacting the amount of energy you're using.

Without explicit recommendations or guidance to do so, the real-time feedback loop instantly influences driving behavior.

These visual cues typically create an immediate effect. You start to drive a bit more sensibly and step down a little less hard on the accelerator. You begin to realize you don't need to accelerate quite so fast to get where you're going. Or, if you're running late, you decide that hitting the gas harder is worth the extra energy consumption. In either case, you can now make an informed decision to use the appropriate amount of energy to get where you need to go based on the environment in which you're operating.

Ask yourself: how can we provide information that teams need to make a better decision?
—Ron Cuirle, Senior Engineering Manager at Target[2]

This real-time data-driven decision enablement is what FinOps is all about. In the data center world, engineers take individual actions that can't easily be traced to their financial impact on the company due to long-term hardware purchases and associated depreciation schedules.

Stories from the Cloud—J.R.

During a previous visit to one of the world's largest cloud spenders, I learned that, despite nine figures a year of annual cloud spend, no one was sharing those costs with the engineers who were incurring them. When they did finally reveal those costs to the engineering team, it was typically 30–60 days later, during an aggregate cost center review. They had the raw data available, but because they hadn't adopted a culture of FinOps, they hadn't implemented a feedback loop directly to those making decisions about how much cloud to use.

Once they did, the results were immediate and dramatic. One of the first teams provided with this visibility discovered that they had been spending over $200,000 per month running development environments that they really didn't need. With only three hours of engineering effort, they shut down those unnecessary resources and saved enough money to hire an additional team of engineers.

The accountability provided by this new visibility revealed something quite interesting. No one gave a specific instruction or recommendation to make a change. Once the engineering manager was able to see the cost of individual environments, she made an informed decision that the team could do without the extra dev environments. She hadn't previously taken action because she didn't have enough information to understand just how large the business impact of leaving the environment running ultimately was. This early FinOps process turned cost into another efficiency metric that engineering teams began to consider alongside other performance and reliability metrics. Engineers tend to be averse to inefficiency and once given the right data (and the leadership support discussed in Chapter 24) tend to optimize cost in the same ways they do other critical metrics.

FinOps is about helping the business make better decisions while moving more quickly. The source of this increased velocity is the frictionless conversations it encourages between teams.

2 Ron Cuirle and Rachel Shinn, "Better Insight: Measuring the Cost of an Application on GCP," Google Cloud Next '19, April 9, 2019, YouTube video, 43:55, *https://oreil.ly/FJ1SP*.

> *What made those teams great is that everyone trusted one another. It can be a powerful thing when that magic dynamic exists.*
>
> —Gene Kim, *The Phoenix Project*

Teams that previously spoke different languages and kept each other at arm's length (e.g., engineering and finance) now build more collaborative relationships focused on what's best for the business. This is FinOps in action. By setting best practices and defining a common lexicon on cloud spending as we'll discuss in Chapter 4, businesses enable productive trade-off conversations to happen.

Core Principles of FinOps

As we have spoken with practitioners who are successfully introducing FinOps culture in their organizations, a common set of values or principles has emerged. Defining FinOps values and ensuring all the process, tooling, and people align to FinOps core principles will help lead you to success. FinOps teams that embrace these principles will be able to establish a self-governing, cost-conscious culture within their organizations that promotes both cost accountability and business agility to better manage and optimize costs while maintaining the velocity and innovation benefits of cloud. These FinOps values are:

- Teams need to collaborate.
  - Finance and technology teams work together in near real time as the cloud operates on a per-resource, per-second basis.
  - Teams work together to continuously improve for efficiency and innovation.
- Decisions are driven by the business value of cloud.
  - Unit economic and value-based metrics demonstrate business impact better than aggregate spend.
  - Make conscious trade-off decisions among cost, quality, and speed.
  - Think of cloud as a driver of innovation.
- Everyone takes ownership of their cloud usage.
  - Accountability of usage and cost is pushed to the edge, with engineers taking ownership of costs from architecture design to ongoing operations.
  - Individual feature and product teams are empowered to manage their own usage of cloud against their budget.
  - Decentralize the decision making around cost-effective architecture, resource usage, and optimization.
  - Technical teams must begin to consider cost as a new efficiency metric from the beginning of the software development lifecycle.

- FinOps reports should be accessible and timely.
  - Process and share cost data as soon as it becomes available.
  - Real-time visibility autonomously drives better cloud utilization.
  - Fast feedback loops result in more efficient behavior.
  - Consistent visibility into cloud spend is provided to all levels of the organization.
  - Create, monitor, and improve real-time financial forecasting and planning.
  - Trending and variance analysis helps explain why costs increased.
  - Internal team benchmarking drives best practices and celebrates wins.
  - Industry peer-level benchmarking assesses your company's performance.
- A centralized team drives FinOps.
  - The central team encourages, evangelizes, and enables best practices in a shared accountability model, much like security, which has a central team yet everyone remains responsible for their portion.
  - Executive buy-in for FinOps and its practices and processes is required.
  - Rate, commitment, and discount optimization are centralized to take advantage of economies of scale.
  - Remove the need for engineers and operations teams to think about rate negotiations, allowing them to stay focused on usage optimization of their own environments.
- Take advantage of the variable cost model of the cloud.
  - The variable cost model of the cloud should be viewed as an opportunity to deliver more value, not as a risk.
  - Embrace just-in-time prediction, planning, and purchasing of capacity.
  - Agile iterative planning is preferred over static long-term plans.
  - Embrace proactive system design with continuous adjustments in cloud optimization over infrequent reactive cleanups.

When Should You Start FinOps?

Determining when to start FinOps has changed considerably since the first edition of this book. A few years ago, the practice typically began once a company had a *spend panic* moment where they inadvertently spent considerably more than they had budgeted. Adoption of the practice has now shifted considerably earlier in the cloud maturity lifecycle. It's now common to see companies at the earliest stages of cloud adoption beginning the practice. This is driven partly by a better understanding of the challenges that cloud billing presents and also a large push by the big three cloud

providers themselves to encourage their customers to get ahead of any cost problems early on. All three have also aggressively matured the cost tools and data provided to allow engineers and architects to consider cost as early as possible when engineering cloud-based solutions.

Unfortunately, many still think FinOps is about saving money or reducing consumption exclusively; therefore, those people tend to believe that the right time to implement FinOps should be measured by the amount of their cloud spend. To be fair, this does make sense on some level. For example, a massive cloud spender could immediately find a lot of potential savings. However, creating accountability for cloud spend and the ability to fully allocate costs are critical components of a high-functioning FinOps practice, which often don't get implemented if you think about it from a retroactive money-saving perspective.

Further, the cultural and skill set changes that FinOps requires from both engineering teams and finance teams can take years to implement. The larger the organization and the more complex their environments, the more they will enjoy compounded benefits from starting the practice earlier.

A successful FinOps practice doesn't require sizable cloud deployments or a multimillion-dollar cloud bill. Starting FinOps early will make it much easier for an organization to make informed decisions about cloud spend, even as its operations are just starting to scale. Therefore, it is essential to understand FinOps maturity; the correct approach to creating a FinOps practice is for organizations to start small and grow in scale, scope, and complexity as business value warrants maturing an activity.

As you'll learn in Chapter 6, it's not possible to install a fully formed FinOps practice from scratch, regardless of your past expertise. It takes time to transform the way engineers work and educate finance people about how cloud operates and to get executives onboard with the right reasons to use cloud. It's a cultural change in ways of working, and the muscle of data-driven accountability and decision making must be built over time.

No one team does FinOps; the central FinOps team works to enable the many various teams around the organization that must work together to manage the business value of cloud. Further, teams across a large organization may be at very different places in their FinOps journey for a long time as each reaches tipping points where individual adoption makes sense.

Although this book can guide an organization to successful practices while helping teams avoid common pitfalls, no organization can proceed directly from zero FinOps to efficient FinOps. Every organization and team must implement FinOps processes incrementally, taking the time to learn from each other as they go.

Like DevOps before it, FinOps is a cultural shift, and the earlier it starts, the sooner an organization will benefit.

Our experience has taught us that you do FinOps from day one, but you engage more of the processes as you scale up. Over the years we have seen companies start adopting FinOps at one of two typical times:

- Formerly, the most common approach was to implement when things went off the rails. In this scenario, spending hits a point where an executive forces a hard stop on cloud growth and demands that a new managing model be implemented. Despite it being a common driver, this is not the ideal way to approach FinOps adoption. Innovation and migrations slow down—or even stop temporarily—during this executive fire drill.
- The wiser approach is taken by executives who have seen the cloud journey play out and align to the FinOps maturity model on *FinOps.org*. The FinOps practice needs to develop at a pace that matches the company's position in the FinOps maturity cycle and the company's maturing use of cloud overall. Initially, a single person might be assigned to manage commitments to cloud providers. They set out to implement an initial account, label, and tagging hierarchy. From there, as the practice gets larger, each part of the process can be scaled up. We now see most new practitioners joining the FinOps Foundation with this adoption story.

But no matter how a company arrives at the decision to implement FinOps, the first critical step is to begin providing visibility of cloud spend to relevant teams in a timely manner, so everyone can begin to understand what's happening and determine how to address cost overruns before they become too large. In a mature practice, cost overruns are prevented via thoughtful architecture design before they even begin. While that level of visibility is achieved, the FinOps team starts educating the larger business stakeholders. As cross-functional teams work together, finance people will learn more of the language of cloud, while engineers begin to grasp relevant financial concepts, and businesspeople can make better decisions about where to invest resources.

The value of starting as early as possible on this cultural shift cannot be overstated, and the benefits of a FinOps practice can be felt and measured almost immediately.

Starting with the End in Mind: Data-Driven Decision Making

One of the most important concepts in FinOps is using *unit economic metrics to enable data-driven decision making by engineering teams*. The idea is to measure cloud spend against business output or value metrics and then take action based on the insights they provide. This is the real-time feedback loop, discussed earlier as the Prius Effect, in action.

Choosing the right business metrics to use for each part of the infrastructure is a complex process that's covered in Chapter 26. For now, the main thing to remember is that providing unit economic metrics to enable data-driven decision making relies on almost every aspect of FinOps, including tagging, cost allocation, cost optimization, connectivity to other finance frameworks, and FinOps operations.

The business metric is important, because it allows you to change the conversation from one that is just about dollars spent to one about efficiency and the value of cloud spend. Being able to say, "It costs $X to serve customers who bring in $Y in revenue" brings a context that helps you make the decision whether $X and $Y are reasonable investments for the organization. Then, as products evolve or change entirely with new features, companies are able to measure the impact of these changes via these business metrics.

The result is being able to determine the difference between good cloud spend and bad—and trend those decisions over time. You should keep business value metrics in mind throughout the book and as you implement FinOps inside an organization.

The nirvana state we are building toward is for your organization to continuously make data-driven business decisions about the value you are getting from your cloud spend.

Stories from the Cloud—Jason Fuller

Jason Fuller, who runs Cloud at HERE Technologies, a multinational organization providing global mapping services, shared a story at a FinOps Foundation meeting that illustrates the point well:

> We had a team way over budget using 9 billion lambda functions a month. I can't influence that directly. But what I can do is sit with the team and understand the quality of the algorithm that you're writing in lambda and determine if it can be tighter.
>
> Do we need that millisecond accuracy you're providing? Yes. OK. The business now understands it's that valuable.

Now we can look at the pricing model for selling the service. We can make a business decision based on how valuable we think the offering is and how much we actually think we can get for it as a product on the market.

As a result, we don't fight about infrastructure anymore—we have a conversation about its business value.

Conclusion

In this chapter we've defined FinOps and described the core principles and values that guide any organization starting the practice to ensure their success as they mature on the long journey of FinOps transformation.

To summarize:

- FinOps is a cultural change that drives collaboration between all teams inside an organization.
- Everyone has a part to play and should become cost-aware: from engineers to finance to procurement to executives.
- Real-time feedback loops on spending encourage continuous improvements to spend efficiency.
- Use unit economic and business value metrics to move beyond talking just about the cost of cloud to making data-driven decisions about cloud investments.
- Implement FinOps in your organization as early as possible and grow incrementally.

Now that you have a basic understanding of what FinOps is, let's look at what the cloud enables inside organizations, and how you can avoid implementing processes that will hamper the good results of a successful FinOps practice.

CHAPTER 2

Why FinOps?

When you look at why organizations use the cloud and the benefits they get by doing so, the importance and necessity of FinOps become obvious. A successful FinOps practice expands and accelerates the business benefits made possible by cloud. Cloud shouldn't be viewed simply as a rehosting alternative but rather as a business accelerator. In the same way, FinOps isn't just about cost optimization, it's about maximizing value.

Use Cloud for the Right Reasons

Cost savings are often touted as the primary benefit of the cloud. But the most successful cloud-first companies have shown the world that scalability and innovation are the true advantages.

Consider Spotify, which uses the scale of the cloud to stream content directly to customers all over the world. Or Flickr, which stores a massive amount of customer data safely in the cloud for secure, fast access. Because of the cloud, these companies compete in a way that they never could with their own data centers. Price is always a factor, but it's a distant third to scale and global availability.

Using the cloud also gives enterprises the ability to move faster and grow their revenue. Even businesses in "nontech" sectors, like financial services, airlines, and retail companies, are turning to software and data to differentiate themselves from competitors. In fact, the top industry represented in the 2022 State of FinOps data was financial services and banks.

Software helps connect businesses with their customers, optimizes their physical assets, and monitors their factories. It delivers the latest pop song, a retail package, and even people themselves all over the globe.

We're no longer an airline. We're a software company with wings.
—Veresh Sita, CIO of Alaska Airlines[1]

As of early 2023, the majority of Fortune 500 companies are massive consumers of public cloud and the big three cloud providers themselves all rank in or near the top 15 most valuable companies in that list. Tech titans such as Microsoft, Amazon, Alibaba, and Tencent sit alongside banking icons like JPMorgan Chase, HSBC, and Bank of America, all using cloud as the primary way of connecting to their customers. Even traditionally nontechnology companies on that list, ExxonMobil and Johnson & Johnson, are accelerating their digital transformations to distinguish themselves from their competitors.

The key components of success are now business agility and speed to innovation. As companies move past thinking of cloud simply as a cost-avoidance strategy, they're increasingly looking toward cloud as their primary driver of innovation. XaaS (anything as a service) allows companies to experiment with more technologies more quickly, while IaaS (infrastructure as a service) provides computing resources at speed and scale never before possible.

Racking servers in a data center and working to engineer already "solved problems" from the ground up is no longer how companies differentiate themselves. Cloud has made the latest technology—scalable infrastructure, machine learning, or IoT (Internet of Things)—available on demand for businesses of all sizes. Complicating efforts to run on-premises data centers, traditional enterprises struggle to compete against the technology scale of Google, AWS, and Microsoft in running infrastructure and attracting the talent to support it.

The enterprises that win align around empowering their engineers to write better and faster code to deliver value to their customers. And those engineers can increasingly get whatever they want, whenever they need it. The cloud has fundamentally transformed their ability to deliver more competitive products.

The cloud value proposition shown in Figure 2-1 from the FinOps Foundation's FinOps Certified Professional course outlines the flywheel effect driving the organization's business outcomes. This is enticing to executives who may feel that their IT function has the potential to drive the organization's digital future and that adoption of the public cloud is a key tool in enabling this vision.

1 Derek E. Weeks, "All Day DevOps: Modern Infrastructure Automation," DevOps.com, August 2, 2017, *https://oreil.ly/K-G4k*.

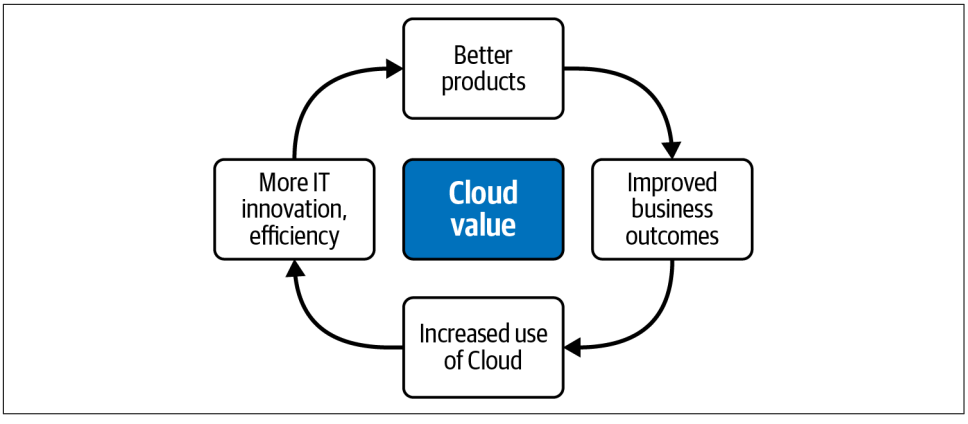


Figure 2-1. Cloud value proposition from the FinOps Certified Professional course

Cloud service providers have truly become a force multiplier for innovation inside any organization—with almost limitless possibilities.

Cloud Spend Keeps Accelerating

When we wrote the first edition of the book, cloud spending had begun to hit a tipping point. In 2019, Gartner had forecast 2022 cloud spend to be around $360 billion, but actual spending in 2022 ended up closer to $500 billion. The latest Gartner forecast predicts that more than half of enterprise IT spending will shift to the cloud by 2025. That's $1.8 trillion that could potentially be moved to the cloud. In our experience, organizations chronically underestimate their cloud spend and industry forecasts are regularly revised upward. Cloud is becoming a material part of organizational IT budgets, impacting both the top and bottom lines for enterprise P&Ls (profit and loss statements). Large enterprises commonly spend in the nine-figures-per-year territory, and we know of a number of individual enterprises whose annual public cloud bills extend beyond a billion dollars.

With all that growth, the time-honored silos between tech, finance, procurement, and business teams have become a problem. To understand why, let's look at our favorite definition of cloud, by Dave Nielsen back in 2011 in his "Cloud Computing Is OSSM" talk. To be called "cloud," Dave said, an infrastructure provider had to be "OSSM":

- On-demand
- Scalable
- Self-service
- Measurable

In the public cloud, the "OSS" in OSSM is what drives incredible innovation while at the same time creating the need for FinOps. Being self-service, scalable, and on-demand allows an engineer to spend company money with the click of a button or a line of code, without going through traditional finance or procurement approval processes. As a result, everyone's role has shifted.

Gone are the days of prepurchasing large amounts of equipment in three- to five-year cycles. In the cloud, the new normal is buying very small units of resources for pennies an hour, making traditional procurement and finance processes ineffective. Engineers aren't getting approval from central procurement teams for each individual cloud resource, and the idea of trying to implement a micro-approval process threatens to slow innovation, one of the main benefits of cloud.

Let's look back at the traditional processes for the data center:

- Equipment was intentionally oversized, with built-in spare capacity to ensure that unexpected growth during the depreciation term could be accommodated.
- If one service used more capacity than was reasonable, it wasn't an issue unless capacity was running low.
- Reducing resource usage wasn't likely to result in any savings, and services that consumed spare capacity often wouldn't cost any more.
- Capacity management was the major cost control used during equipment lifecycle. When capacity was running low, the services resource allocation would be reviewed and adjusted.
- The cost of the equipment was paid up front, with possible well-understood monthly costs like data center fees, software licensing, and so on.
- Costs were reported and reviewed monthly or even quarterly. Infrequent reporting was acceptable, because the day-to-day costs of the data center did not vary.

And now let's look at the cloud in comparison:

- There is no up-front equipment to purchase, and spare capacity is always available. Prepurchasing capacity isn't usually required, and companies can save by not paying for capacity when it isn't needed.
- When services use more resources than they need, the result is higher running costs. Reducing the size of the resources allocated to services results in realized cost reductions.
- Due to the availability of capacity from cloud service providers, capacity management is not a major concern. Removing this process means services are no longer artificially capped in resource availability.

- Resources can be consumed during busy periods and then removed for slower times. This variable consumption model results in lower operational costs but also makes predicting costs tricky.
- Billing is no longer simple to understand, as resources are individually charged in micro amounts.

Reviewing costs on a quarterly or even monthly cadence often results in sticker shock when unexpected or forgotten resource usage adds up to material levels in the bill.

The dramatic shift from fixed to variable spend changes the way you must report costs. A set of restraints has been removed, fundamentally altering how companies build and deliver software, for better or worse. Managing the variability of that spend is a very different job today than it was when IT ran its businesses based on dependable fixed costs.

The Impact of Not Adopting FinOps

Within the data center there is an artificial limit to the available capacity engineering teams can quickly access. Increases in cost are tightly controlled by business processes that are used to review and approve any extra spend. This limits the speed at which your costs can change and how quickly teams are able to experiment and innovate. The cloud removes the limitation on available storage and compute and empowers your engineering teams to get quick access to the resources they need. It isn't possible for your engineering teams to call an API that provisions hundreds of new servers into your data center, which you would then need to pay for. In cloud, engineers do this all the time.

When you're focused on speed and innovation, it's easy for cloud bills to soar. In response, companies too often clamp down on cloud, causing innovation to slow, and threatening to make them less competitive in the process.

A McKinsey study published in September 2020 titled "How CIOs and CTOs Can Accelerate Digital Transformations Through Cloud Platforms" concluded that companies that reap value from cloud platforms treat their adoption as a business-technology transformation by doing three things:

- Focusing investments on business domains where cloud can enable increased revenues and improved margins
- Selecting a technology and sourcing model that aligns with business strategy and risk constraints
- Developing and implementing an operating model that is oriented around the cloud

However, they also found that cloud economics, skills, processes, and organizational changes required are too complex and span too many different parts of the business for infrastructure heads to manage on their own, resulting in an overwhelming majority of large institutions experiencing one of the following failure modes:

Pilot stall
: Value proofs of initial cloud workloads or migrations come up short, preventing a successful business case to extend the use of the cloud.

Cloud gridlock
: Cloud initiatives become jammed up in queues because IT cannot build out the automation or reference architectures required to use public cloud platform services in a secure, resilient, and compliant fashion.

Lack of value from "lift and shift"
: Support for cloud collapses after migrations that don't take advantage of cloud optimization levers or native technologies, which results in inadequate value returns for the cost.

Cloud chaos
: A lack of guidance from leadership leaves engineers to their own devices in configuring cloud services, leading to cost, security, and compliance issues.

McKinsey concluded that, as a result, although spending is growing quickly, only 10%–15% of enterprise workloads have successfully transitioned to cloud from the data center.

Investment in FinOps allows FinOps capabilities to be developed and leads to better financial management of cloud services. This ensures cloud initiatives deliver measurable business value, and each of these value-driven initiatives gives the cloud value flywheel another push.

Extending the cloud value proposition from Figure 2-1 by adding FinOps is shown in Figure 2-2. To sustain investment in FinOps, it is essential to show measurable improvements in cloud value. Quick wins are necessary until the organization develops confidence in FinOps and the business value of the cloud. Every time a FinOps initiative results in improved business value, it gives both the cloud value flywheel and the FinOps value flywheel a further push, providing that it is communicated to the stakeholders that are investing in the cloud and FinOps. If these improvements in business value are not communicated, they don't exist, and cloud adoption across the company can stall.

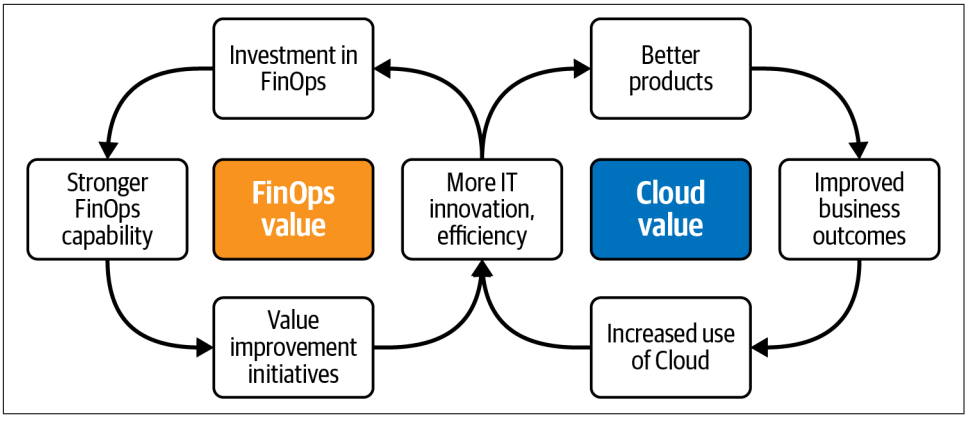


Figure 2-2. Organizations should sustain their FinOps investment as long as it is justified by improvements in cloud business value (from the FinOps Certified Professional course)

Informed Ignoring: Why Start Now?

Before covering *informed ignoring* within FinOps, let's talk about simply *ignoring* FinOps. Ignoring happens when you're not doing any measurement of spend efficiency. You (or your leadership) have decided you do not need to focus on cloud costs *yet*. The concern is that a FinOps practice will slow things down. You just have to get the migration done or release a set of features, whatever the cost.

The scenario usually goes like this: the business decides it has a big pot of money to get a migration done, say $50 million. Statements like this are tossed about: "Just get the engineers to get it done and we'll worry about cost later." The business leaders think there is no way they'll ever hit such huge cloud spend numbers, especially at the beginning when the monthly bills go from $5,000 to $50,000 to $250,000 to $1 million. In comparison to other IT costs at a large enterprise, they're all still small numbers.

So the engineers keep toiling away at the migrations, and along the way they build up a lot of inefficiency. At some point, the cost will begin to outstrip the benefits and become out of sync with expectations of the business. This was the norm when we published the first edition of the book, and invariably companies would start FinOps only after a spend panic moment forced them to do so.

Thankfully, we have seen a shift over the past couple of years, with more informed practitioners setting up a FinOps practice at the same time as the transition to the cloud. In those scenarios, we increasingly see an *informed ignoring* approach. There is a focus on understanding costs from the beginning, ensuring a comprehensive cost allocation strategy is in place, reporting on big cost drivers, monitoring anomalies, and exploring opportunities to increase cost efficiency. But actions to reduce cost

aren't necessarily taken during this period; rather, there is active monitoring and forecasting to make sure spending is still in line with the business's expectation.

Conversely, in taking a simple *ignoring* approach, no one pays close attention to the spend except when they are paying the bills. In an *informed ignoring* approach, the business is consciously watching the costs increase, forecasting their growth, and analyzing saving opportunities to see when they hit a point where the comfort level of the business forces action to adopt additional domains of the FinOps practice such as usage or rate optimization, as discussed in Part III.

Critically, work is being done to prepare everyone to do active FinOps: reporting and visibility are being put into place, cross-team communication strategies are implemented, and complicated processes like forecasting cloud spend are being honed while the engineers stay focused on speed at all costs. At some point, the forecasts indicate that they are going to be spending past the point of comfort by a certain date, but in an *informed ignoring* approach, that date is months away, leaving time to gently change the course of the organization rather than requiring an aggressive clampdown on cloud spend, which could ultimately hamper cloud adoption and derail larger business objectives.

This is the soft hand of FinOps: keeping the engineers aware, upleveling finance to deal with the new complexity, and allowing executives to make informed choices while there is still time. With this visibility, engineers tend to start considering costs over time as they deploy infrastructure. They become aware that if waste becomes too high, the business will eventually ask them to fix it. The feedback loop of spend efficiency begins to germinate.

In Figure 2-3, Forrest Brazeal cleverly illustrates the decreasing value and growing dangers of a lift-and-shift cloud migration. Organizations that take the quick wins of a lift-and-shift migration, but then don't follow up with the steps forward into cloud native, find the positive value created by the lift-and-shift becomes overwhelmed by its negative side effects and cloud value is eroded simply by the passage of time. Measuring and making decisions based on the value derived from cloud spend helps an organization to maintain the value it gets from the cloud while investing in reducing the dangers in its cloud migration.

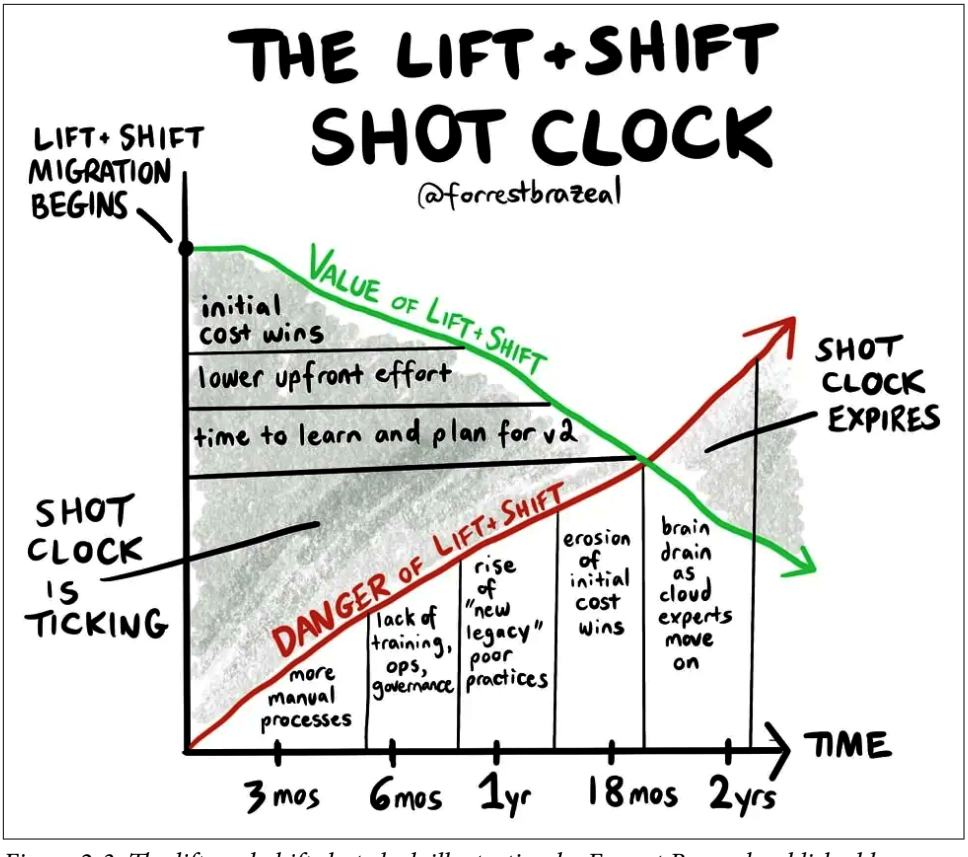


Figure 2-3. The lift-and-shift shot clock illustration by Forrest Brazeal published by ACloudGuru, February 2020

The decision made by business leaders to ignore opportunities is not made once but repeated as your journey in the cloud continues. As the threshold of comfort with the spending level or size of the reduction opportunities becomes significant, your business leaders may change the decision to ignore and start allocating priority to FinOps activities. Without you keeping your business leaders up to date about the size of spending and reduction opportunities, they will grow unchecked and be out of sync with expectations and eventually come as a surprise.

The core concept of informed ignoring is that it is OK for your organization to choose not to spend time on reducing your cloud spending. But there needs to be awareness within the business, especially among finance and business leaders, to decide to ignore them.

Finally, there are some decisions that should never be ignored. It's really hard to go back and reverse engineer a comprehensive tagging/labeling/account methodology—

more on this later in Part II of the book—after you have millions of dollars of cloud spend. As you can see in Figure 2-4, the 2022 State of FinOps data showed that cost allocation was considered the most important FinOps capability. If you bypass the foundational allocation capabilities, you're ultimately unable to build the reports that would enable you to do informed ignoring in the first place, leaving simple ignoring as the only option.

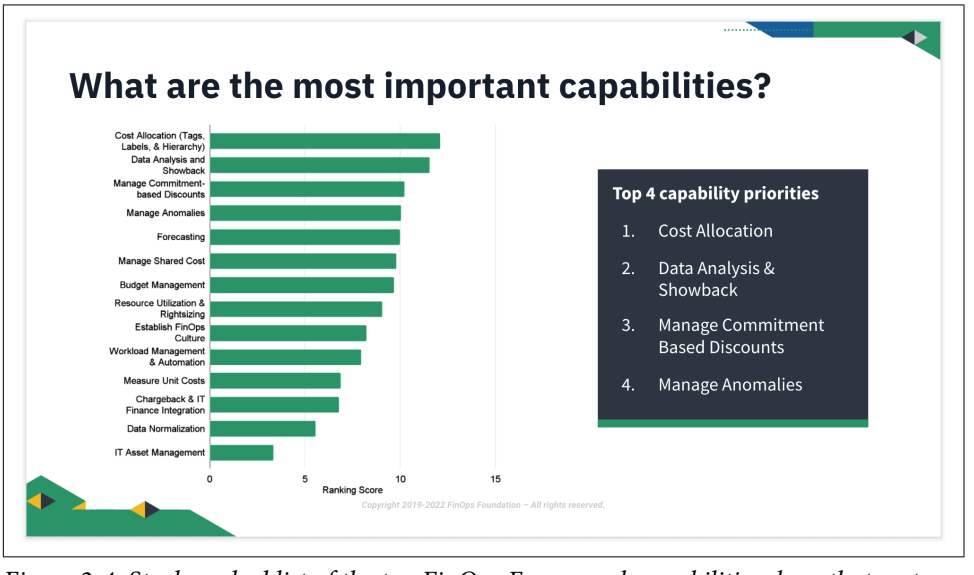


Figure 2-4. Stack ranked list of the top FinOps Framework capabilities show that cost allocation and data analysis and showback are the most important parts of a FinOps practice (source: 2022 State of FinOps report)

Even if you don't have the time or resources to implement an active FinOps practice, get started now with the practice of allocating, tracking, and forecasting your cloud spend so that you're ready to optimize when you need to.

Conclusion

The primary advantage of cloud is the speed of delivery and innovation, not cost savings.

To summarize:

- Cloud spend has—or soon will have—a major effect on organization balance sheets.
- A lack of investment in FinOps early on can cause cloud adoption to hit headwinds or stall.

- The procurement team no longer has control of the spending. In the cloud, this power has been pushed to engineers.
- FinOps allows you to operate at the per-second speed of the cloud rather than relying on traditional monthly or quarterly spend reviews, which allows you to avoid unexpected costs.
- Start with an *informed ignoring* approach before you are ready to perform all the domains of FinOps for an active FinOps practice.

FinOps is brand new to most companies, and while it doesn't have to be done all at once, it must ultimately be implemented and adopted throughout the entire organization to be successful. The next chapter examines the culture of FinOps and the roles everyone must play to enable a successful FinOps practice inside an organization.

CHAPTER 3

Cultural Shift and the FinOps Team

FinOps is more than a technology solution or a checklist handed off to a team. It's a living, breathing way of maximizing the value of cloud. Technology certainly helps solve problems like monitoring optimization recommendations and detecting anomalous spending, but it can't have a conversation about the business value of those actions.

In this chapter, we'll look at who drives FinOps and where they tend to sit in an organization. We'll examine FinOps personas and roles, because ultimately it's people who make FinOps work.

Company culture around spend must evolve alongside tooling and processes in an organization. There are plenty of obstacles and challenges in implementing a successful FinOps practice. They'll either be caused by or solved by people and teams. Ultimately, it comes down to you. Will your teams develop and adopt the right principles and practices for successful FinOps?

Deming on Business Transformation

While researching for the second edition of this book, we came across key principles for managing business transformation published in 1982 by renowned engineering and business professor W. Edwards Deming. Many of the key points of FinOps echo Deming's principles, so it's worth looking at a couple of Deming's key principles.

Deming focused on constancy of purpose. His view was that the best way to drive a company to long-term success was to foster long-term thinking and be constantly implementing continuous improvement.

He also stressed the importance that all teams involved in a transformation need to work as a single unit, breaking down the barriers between departments to enable the whole organization to identify problems early on and collaboratively work toward solutions. Likewise, a successful cloud and FinOps transformation requires each team involved in the management of cloud to understand that the transformation is everybody's job.

Leadership and education feature heavily in Deming's principles. He did not recommend that leadership manage primarily via targets and numbers, but to manage the system in a way that allows staff to contribute with a pride in their work. In Chapter 24, we'll discuss the related concept that leadership support for FinOps is not optional and that mandates to reach arbitrary targets will ultimately fail. Rather, leadership needs to support and drive a cultural shift that empowers teams to make decisions that balance the cost versus the value of cloud.

Deming also added that massive training is required to instill the courage to break with tradition, and that every activity and every job is a part of the process.

The FinOps transformation of an organization closely mirrors Deming's principles four decades later. We encourage you to read all 14 of Deming's principles published in *Out of the Crisis* (MIT Press). You may find it helps to reference Deming's work when pitching FinOps to your organization.

Who Does FinOps?

From finance to operations to developers to architects to executives, everyone in an organization has a part to play. Take a look at the wide variety of job titles quoted throughout this book, and you'll see what we mean.

> *It's everyone's job to help the company go faster. In an agile world, we are all servant leaders to help the engineers deliver what they are doing. Everyone else should be there to unblock delivery.*
>
> —David Andrews, Senior Technology Manager, Just Eat

Whether it's a small business with only a few staff members deploying cloud resources or a large enterprise with thousands, FinOps practices can and should be implemented throughout the organization, albeit at different levels of required tooling and processes and at different cadences depending on the value delivered in each place.

So what skills and roles does a FinOps team need to have? Later in the book, as we work through the *optimize* phase, we'll go into ways a FinOps team generates recommendations that need to be considered by the wider organization, such as changing resource configurations or making commitments to cloud service providers. While

these will undoubtedly save the organization money, trust is required between product, finance, and engineering to carry them out.

To create this trust as quickly as possible, the FinOps team must have cloud expertise. Without it, recommendations might be wildly incorrect, or engineering teams may waste valuable development time trying to explain their way out of making necessary performance modifications. A FinOps team with cloud domain expertise builds trust and credibility.

FinOps must have executive support, be capable of sharing best practices, and use a common language that ties disciplines together, as we'll discuss in the next chapter. Not only does this help to spread the message that FinOps is important to the company, but it also ensures that staff will have time allocated to perform required tasks. When done well, FinOps doesn't increase conflict between technical and business teams. Instead, that conflict is removed. This is accomplished when teams use a common language that helps create alignment around business goals, thus making sure that everyone understands and supports those goals.

A central FinOps enablement team is the driver of a FinOps mindset across the organization, evangelizing best practices for incorporating cost considerations into development plans. FinOps practitioners perform some centralized tasks like managing and reporting on commitment-based discounts to the cloud providers or centralizing reporting to make sure everyone is looking at the same numbers. This prevents other teams from reinventing the wheel while taking advantage of economies of scale in relation to commitments.

However, the actual implementation of many FinOps best practices must be decentralized to the engineers inside various product and application teams across the company. A good example of this is rightsizing and idle usage reduction. While a central FinOps team may help with data and goal setting, engineers in various lines of business must make data-driven decisions about when to make changes to the infrastructure based on business value and level of effort.

> *When you can give finance an alert about a spend spike in 24 hours as opposed to a quarter later, it buys a lot of credibility. If it takes you multiple months to figure out what happened to cause an overrun, it won't give confidence to finance that tech knows what they are doing.*
>
> —Joe Daly, formerly Director of Cloud Optimization, Nationwide

Another vital role for the centralized FinOps team is to facilitate the conversations among teams and to foster trust. Finance teams must partner with engineers, using shared data and reporting, to enable everyone to quickly find and solve the situations that need addressing.

Armed with the reporting and billing knowledge from the FinOps team, engineers will be able to show where, when, and why a plan or cost exceeds budget. This shared knowledge builds confidence, trust, and efficiency.

Instead of a focus on centralizing spend approvals, FinOps builds visibility of the spend to the appropriate areas to create accountability. The cloud-aware expertise of the FinOps team allows the other teams to understand how each specific billable item can be distributed into chargeback and showback. These are terms used for how you display and handle costs. We'll dig into this subject more in Chapter 11.

Why a Centralized Team?

We have so far referred a lot to the FinOps team and to the centralized functions a FinOps team performs. It's worth pointing out here that we'll continue to refer throughout the book to a centralized FinOps team, but there are a number of different FinOps Team models that work. How your FinOps team is structured will largely depend on the type of organization you have and the way it will support a cross-discipline, coordination machine that the FinOps team represents.

While we will still refer to the FinOps team, interpret that as the coordinating body that drives FinOps best practices, but understand that whatever organizational design that takes, there are things that a FinOps team should do organization-wide to make the transformation as successful as possible.

The unbiased central team, with ties to business leaders, engineering teams, and finance teams, shares objective best practices and recommendations. The members of this team are never seen as pushing a specific agenda to benefit themselves, which builds more trust in the advice they give. Critically, they must also work to incorporate business objectives for each cloud workload.

When this central group drives the rules of cost allocation and pushes out clear communication about how teams are doing, everyone is reassured that they're being managed and measured against the same criteria. If one of the budget-holding teams drives the process, which happens when a group is responsible for the lion's share of cloud spend, it can raise questions about the integrity of information. If cost allocation is done independently by each team, there will inevitably be overlap or gaps in spending data. In either case, doubt grows. And when teams lose trust in data, they can't be held accountable to it.

When individual teams try to build out their own cloud reporting processes, disagreements arise about whose cost was whose. An organization-wide FinOps team solves this issue, fostering agreement that the spend data is the correct data and then objectively attributing each cost to the appropriate engineering group.

Finally, the central team defines what the organization uses as a business metric. In fact, this is a key early task of the FinOps team. A dollar is not just a dollar when it comes to cloud spending. One can choose to look at cost amortized or not amortized, and with custom rates applied or excluded. Shared costs of an application from another service may be included or invisible. When the business metric used across

teams is clearly defined, everyone speaks the same language. In the next chapter, we'll dive deeper into the language of FinOps.

The FinOps Team Doesn't Do FinOps

It's easy to assume that the daily work of FinOps can be assigned to a central team and siloed from the work of the rest of the business. But FinOps isn't only happening inside your FinOps team; they are primarily an enablement team that supports the rest of the organization with subject matter expertise in the very real complexities of cloud billing data and pricing models, centralized data and reporting, commitment management, and related functions while helping the distributed engineering teams with the data they need to make daily decisions about their infrastructure. It should not be expected to be a central team for managing infrastructure or performing the bulk of optimization activities. That responsibility is pushed to the groups that understand their own infrastructure and their business objectives the best. The FinOps team aims to propagate repeatable patterns throughout the organization to accelerate FinOps practices at the edges.

> *The other teams in our organization ask for me to tell them about FinOps and explain how we are going to protect them from overspending. To which I say, I'm not, I'm gonna help you protect yourself.*
>
> —Alison McIntyre, Cloud FinOps at Lloyds Banking Group on *The FinOpsPod*, 2022

A FinOps team will do more of the work early on in an organization's cloud maturity, as everyone becomes familiar with their new responsibilities. But as time goes on, teams will become more self-sufficient, and the FinOps team will be able to tackle fewer of the routine tasks and focus on more complex projects related to cloud efficiency.

> *Why is subject matter expertise in cloud billing needed? As an example, AWS's S3 storage service has at least 6 different types of "actions" you can get charged for—storage pricing, request and data retrieval pricing, data transfer and transfer acceleration pricing, data management and analytics pricing, replication pricing, and the price to process your data with S3 Object Lambda. There are also at least 8 different storage types from Standard to Infrequent Access to Glacier to Deep Archive. That is 48 combinations of pricing options for a single service. Amazon currently has hundreds of different services all with different pricing options. Subject matter expertise in a FinOps team can help navigate this complexity.*
>
> —Ashley Hromatko, formerly Director of Cloud FinOps at Pearson

FinOps teams also might, as they mature, pick up more responsibility for automating the really routine tasks that aren't a result of engineering teams being inefficient. There is a certain amount of technical debt introduced when moving to cloud, such as putting storage in a more expensive tier than it needs to be or purchasing higher tiers of service than required, that the FinOps team might be empowered to fix on its

own, once the FinOps team has established the credibility to do so and the trust with engineering teams.

This enablement pattern is not a new concept; consider security in the cloud. It's a shared responsibility model. Just as the security team in an organization is not solely responsible for security, the FinOps team is not there to optimize all the things. It's there to provide the resources for the engineers closest to the cloud resources to decide on the right optimizations at the right time, or to not optimize them using *informed ignoring*, covered in Chapter 2.

The work that your central FinOps team does enables and accelerates your organization to use cloud more effectively. It doesn't replace the daily duties that other engineering teams and business leaders have to continuously optimize their own infrastructure.

The Role of Each Team in FinOps

Figure 3-1 demonstrates how organizations operate in the FinOps model. A cross-functional team, such as the Cloud Center of Excellence (CCoE), manages the cloud strategy, governance, and best practices and then works with the rest of the business to transform how the cloud is used.

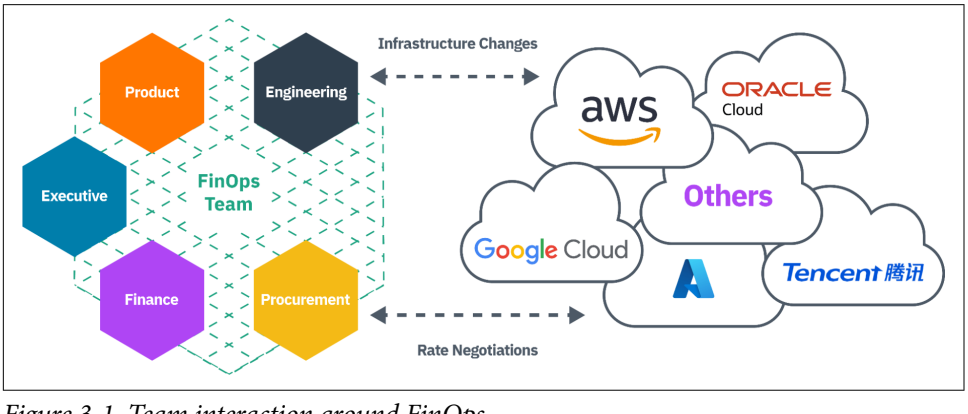


Figure 3-1. Team interaction around FinOps

Individuals at every level and in every area of an organization have different roles to play in the FinOps practice. These include the following.

Executives and Leadership

Executives (e.g., VP/Head of Infrastructure, Head of Cloud Center of Excellence, CTO, or CIO) focus on driving accountability and building transparency, ensuring teams are being efficient and not exceeding budgets. They're also drivers of the

cultural shift that helps engineers begin considering cost as an efficiency metric, discussed in Chapter 24.

Engineering and Developers

Engineers and ops team members—such as Lead Software Engineer, Principal Systems Engineer, Cloud Architect, Service Delivery Manager, Engineering Manager, or Director of Platform Engineering—focus on building and supporting services for the organization. Cost is introduced as a metric in the same way other performance metrics are tracked and monitored. Teams consider the efficient design and use of resources via activities such as *rightsizing* (the process of resizing cloud resources to better match the workload requirements), allocating container costs, finding unused storage and compute, and identifying whether spending anomalies are expected.

Finance

Finance team members use the reporting provided by the FinOps team for accounting and forecasting. They work closely with FinOps practitioners to understand historic billing data so that they can build out more accurate cost models. They use their forecasts and expertise from the FinOps team to engage in rate negotiations with cloud service providers.

Procurement and Sourcing

Procurement and sourcing people manage the business's relationship with vendors, including the cloud service providers. This relationship may be very different from other IT services relationships managed by the company in the past, and it may be more complex and varied to manage. They are interested in getting the best pricing for the commitments the company is making to each vendor, and making sure that the organization receives all the benefits of its patronage and commitments over time.

Product or Business Teams

Product teams members, including Product Managers, Product Owners, Portfolio Owners, Service Owners, Application Leads, and the like are responsible for a service or product or product line. They will work closely with the FinOps team to understand, often in ways they could not when supported in the data centers, the total cost of running the features of their product or application. Product leads are interested in new features to solve customer needs, and will be able to use FinOps-provided data to understand product profitability, direct cost efficiency efforts, and more accurately forecast costs based on new feature releases.

FinOps Practitioners

> *The finance people see me as a techie. The tech people see me as a finance person.*
> —Ally Anderson, Business Operations Manager, Neustar

FinOps practitioners are the beating heart of a FinOps practice. They understand different perspectives and have cross-functional awareness and expertise. They're the central team (or person) that drives best practices into the organization, provides cloud spend reporting at all the needed levels, and acts as an interface between various areas of the business. They can be cloud-savvy financial leaders or cost-conscious engineers.

A New Way of Working Together

Each of the previous functions needs to integrate like never before. Some say engineers have to think a bit more like finance people and finance people have to start thinking like engineers. That's a great start, but the entire organization needs to shift from a centralized cost control model to one of shared accountabilities. Only then can the central FinOps team empower the organization to move and innovate faster.

This cultural shift also enables those in leadership positions to have input into decision making in a way they currently don't. Based on leadership input, teams make informed choices about whether they are focused solely on innovation, speed of delivery, or cost of service. Some teams go all-in on one area with a growth-at-all-costs mindset. Eventually the cloud bill gets too big and they have to start thinking about growth and cost together. For example, "Move fast, but keep our cost per customer transaction below $0.45."

Where Does Your FinOps Team Report?

More often than not, FinOps teams report to the technology leader, but they can also report to finance or other functions. The State of FinOps survey asks where FinOps teams report, and as you can see, the results in Figure 3-2 show that FinOps teams report to many locations within different organizations. Most commonly, the FinOps team is aligned within or alongside technology teams.

Figure 3-2. Most common executive FinOps reports to, from the 2022 State of FinOps report

This means it is typically part of the CTO or Head of Technology function. Some organizations also place it within a CIO function, though this is declining over time. Having the technology leadership watch the technology spending may be akin to the fox watching the hen house since the group spending the money is also the one monitoring it. However, there are numerous advantages to having FinOps sit in the technology organization. Initially, more weight and credibility are given to its recommendations by engineering and tech ops teams, who might be more dubious about recommendations coming from a finance team. Most importantly, the technology teams are the ones who best understand the complexities of cloud and are able to most quickly set up the practice of managing highly variable cloud spend. Finance-driven FinOps often initially attempts to apply existing financial reporting and forecasting systems not designed for a cloud bill, which may have millions of individual charges each month that change daily at variable usage and rates. Increasingly, we're seeing a hybrid function emerge where there is a COO or CFO

for Technology under the CTO. A story from Equifax demonstrates this structure in Chapter 6.

Regardless of where they sit, the central FinOps team must partner closely with the finance and engineering organizations. Small companies might not have a dedicated place in the organization for a FinOps team. A common pattern in that case is to assign a single person or split part of an individual's time to practice and promote FinOps within the business. A large, complex spender may have 10 or more people in the central FinOps team.

Understanding Motivations

> *Trust usually comes when there is empathy for other people's problem domain.*
> —Sascha Curth, Head of Cloud Economics and Governance at OLX

Whenever a disparate group of smart people come together from different parts of an organization, there will invariably be some friction, largely because they have different goals and targets. Finance focuses on cost efficiency and ensuring budget targets aren't exceeded, while operations teams think about offering high-quality service with minimal service issues and outages.

Let's take a deeper look at the motivators of various personas involved with FinOps.

Engineers

- Want to work on hard problems that are meaningful and challenging
- Want to deliver software quickly and reliably
- Hate inefficiency and want efficient use of resources
- Stay up to speed on the latest tech
- Are measured by performance, uptime, and resilience
- Want to deliver features, fix bugs, and improve performance

Usually, engineers are incentivized to get the latest software features or bug fixes to customers instead of on the finance side of things.

FinOps teams tend to focus on asking engineers to help them understand costs, review their environments for potential optimizations, and determine how cost-efficient they are.

Jason Fuller, Head of Cloud at HERE Technologies, follows this approach with his engineers:

> Let my team and the FinOps organization do that for you. Let me set a storage lifecycle policy for you. You recognize, as an engineer, that you should have one, but it's number 100 on a list of 100 things. So I'll take care of that. The strategy that works best with them is "Let me help you. I'll take care of all that. Let me standardize and write your storage lifecycle policies so you don't have to."

In some cases, a FinOps practitioner will spend time performing these repeatable tasks in partnership with engineers, highlighting data to surface potential areas where engineers are able to optimize and then helping them by removing—and standardizing—processes that can be centralized. Sometimes the FinOps team is pushing engineering to take on this task of optimizing themselves as early as possible, and other times it has the credibility to take on some of the automated cleanup of tech debt for them.

Finance People

- Want to accurately forecast and predict spending
- Want to be able to charge back and/or allocate 100% of spending
- Seek to amortize costs appropriately to the teams responsible
- Want to split out shared costs, like support and shared services
- Want to control and reduce costs, but maintain quality/speed
- Want to help executives inform cloud strategy
- Want to be aware of budget risks ahead of time

Finance people can experience sticker shock when the organization first adopts cloud. Changes to the spending approval process, moving from capital expenses with their predictable depreciation schedules to operating expenses, highly variable and less predictable spending, and the crush of cost and usage information can all be disorienting. They're sometimes not sure how to keep up with the rate of change and don't know if they can trust the numbers coming from the cloud provider billing data.

Thus it's important to remind traditional finance teams that cloud fits a model they already understand. It's simply a consumption-based service like utilities. It just happens to have 100,000 SKUs, it moves in microseconds, and its spending can go out of control over the course of a weekend. It is, of course, a different sourcing process and granularity, but it's still the same financial model. However, for a finance person new to cloud, it's easy to get intimidated.

Executives and Leadership

- Want to drive shared accountability to teams
- Desire a digital business transformation
- Want to shorten time to market for new services
- Seek a competitive advantage
- Want to establish a successful cloud strategy
- Need to define and manage KPIs (key performance indicators)
- Must prove the value of tech investments

Chapter 2 showed how organizations use software and internet-connected technologies to differentiate themselves from their competitors. The cloud is one of the primary tools to accelerate this digital change. Executives are setting their cloud-first strategies and leading their teams into the cloud. As cloud spend becomes material within an organization, it's essential for these same executives to drive the importance of tracking and balancing costs for the engineering teams. Executives support the cultural change that FinOps practitioners are working to implement within the organization. From the top down, leadership provides guidance on the prioritization balance between good, fast, and cheap. For FinOps to work well, leaders and executives must also be aligned with one another—across disciplines and vertically across levels—on the goals of cloud, and the controls they want to govern its use.

Procurement and Sourcing People

- Traditionally measured on discounts achieved during negotiations
- Wish to ensure spend is in line with vendor agreements
- Want to build relationships with strategic vendor-partners
- Want to negotiate and renew vendor contracts

Procurement is no longer the gatekeeper of IT spend. With engineers directly setting up cloud resources, procurement has become a world of distributed responsibility. Maintaining visibility into cloud spend and generating accurate forecasts becomes more important so this data can be used to drive vendor relationships and contract negotiations. As procurement teams embrace FinOps, they don't force some micro-approval processes on cloud spend. Instead, they choose to help drive accountability while enabling teams to get access to the right resources for innovation.

FinOps Throughout Your Organization

It's no longer acceptable for teams to consider only their own priorities. If the technical operations team doesn't take into account the impacts of their cloud spending, finance will have an impossible challenge of forecasting and budgeting an organization's cloud spend. Alternatively, if finance takes full control of cloud spend and requires approvals for every resource, the organization will struggle to take advantage of the speed and agility of having variable spend resources and on-demand infrastructure. Remember, the advantage of the cloud is the speed of innovation.

Stories from the Cloud—Rob Martin

Rob Martin, Director of Learning at the FinOps Foundation, shares this anonymized story from recent FinOps Certified Practitioner training courses about the way the FinOps personas work together:

> As the FinOps culture of accountability gets established across the organization, all of the disciplines begin to work more closely together. When the product manager identifies a new feature, they can quickly identify the revenue or customer impact of offering it, and the margin they'd be on the hook to produce. Leadership responsible for the profit and loss statement where the product lives can okay the investment and has the flexibility to adjust budgets for the teams that have to develop, so they lay out the success parameters, what they expect it to cost and to sell. This gives engineering leadership the clear direction to build a plan, schedule the work out into sprints to create the minimum viable product (MVP), and estimate the cost curve over the time to build which finance had visibility to. Everyone can circle back as that plan solidifies, the adjustments are made to budgets, resources get requested, with the appropriate tags that we're all in on. In the old days, this process would take months to brief, to coordinate, to get approvals on; now we can do it in days or weeks.

Hiring for FinOps

It's important to note that you can't facilitate a cultural shift to FinOps just by hiring practitioners or bringing in a contractor. With FinOps, everyone in the organization has a part to play. A FinOps requirement should be added to all hiring, from executives to finance to operations, such as listing FinOps—or more specifically the management of the value derived from cloud—as part of each job description or providing FinOps training as part of onboarding new hires into the organization. Ask engineers how cost metrics play a part in good service design, and ask finance how the variable spend of cloud changes normal finance practices. Without hiring new talent with a FinOps understanding or ongoing training for the new members joining your teams, the culture you've built will slowly be eroded by new staff lacking the correct mindset.

Roles will evolve to meet the needs of FinOps. More poly-skilled employees who have a business head, fiscal thinking, and technology acumen will emerge. Finance people will learn cloud, just as IT people will learn finance.

Think back to when the idea of full-stack engineers was new; now it's time to start thinking about full-business people.

Here are some of the common skill sets that you'll need to fill when building out your FinOps practice:

Policy governance
: Creates policy documents, standards, key performance indicators (KPIs) and objectives and key results (OKRs), and governance frameworks to guide the organization's cost allocation and cloud use.

Technical writing
: Creates documentation about FinOps processes, helps socialize standards (tagging), and sends budget alerts.

Analysis
: Digs into cost abnormalities, learns about cloud cost models and explains them to engineers and to finance, and creates and delivers reports to executives.

Engineering
: Considers the costs of architectural decisions and assists in automation of billing data, optimizations, reporting of budgets and forecasts, and governance.

Automation engineering
: Focuses on automating cloud tooling, either automating the management of cloud billing mechanisms like budget alerts and commitments, or automating optimization recommendations.

Data engineering
: Builds systems that collect, manage, and normalize raw data into usable information for data scientists and business analysts to analyze and report on. Data engineers implement methods to improve data reliability and quality.

Training
: Gets your teams up to speed on what FinOps means for them; requires training materials to be made available and management of courses either self-paced or in person.

Evangelism/marketing
: Builds the profile of FinOps within an organization; requires effort to build a promoter base, while winning over the detractors by helping them understand the value that FinOps provides.

The preceding skill sets may initially be covered by a single jack- or jill-of-all-trades but over time, as the practice and spend grows, they will likely be split across a larger number of specialists. Chapter 6 talks about common team sizes based on maturity and lays out some sample team structures.

FinOps Culture in Action

As an example of how a FinOps culture, along with the language of FinOps, can enable the business, let's take a look at how the introduction of containerization impacts cost visibility. With the introduction of containerization, operations teams are able to pack more services onto the same compute resources. Services such as Kubernetes give control and automation to operations teams like never before. It's easy to see that for operations teams, implementing containerization is a great opportunity. And it has played out across the industry in the large-scale adoption of container environments.

You might think that having more services on the same compute resources would mean more efficient costs, but those benefits can come with a loss of visibility. Cloud billing data has the underlying compute instance costs, but there may be no details to help finance work out what containers from which applications are operating on top of each instance.

Consider how this plays out without a FinOps culture.

The finance team will understandably want to know how to split out the costs of the underlying compute resources to the correct business units. So they ask the engineers for help. This request means that engineers must stop what they are doing and shift their focus to costs and financial data. Because that shift will ultimately result in a lack of productivity—the finance team might conclude—and then start championing the idea to executives, that containerization is bad for business operations.

However, when you apply FinOps and introduce FinOps practitioners to the conversation, you end up with a different outcome. The practitioner will have deep knowledge of what data is or isn't available from examining the cloud service provider's billing files. Finance will learn and understand the basics of containerization and why it benefits the business. Meanwhile, the operations team learns about the importance of chargeback and showback.

When finance asks for the cost breakdowns for the containers running on the cluster, a FinOps practitioner will understand the finance team's perspective: they see only the overall cost of the cluster. On the other hand, while engineering teams know which container is scheduled on each cluster instance, they can't easily associate this data with the cloud bill. But when they share that information, the FinOps practitioner can take on the burden of working out the costs.

The FinOps team then takes this allocation data and performs the needed analytics to combine it with the billing data. Now finance gets the reports they need, and already busy engineering teams can keep working. Because of this cross-functional approach, finance can now see containerization as an enabler of efficiency, not as a blocker.

Difficulty Motivating People Is Not New

For the last two years, in the State of FinOps survey the number-one challenge FinOps practitioners face is motivating engineers to take action, as shown in Figure 3-3.

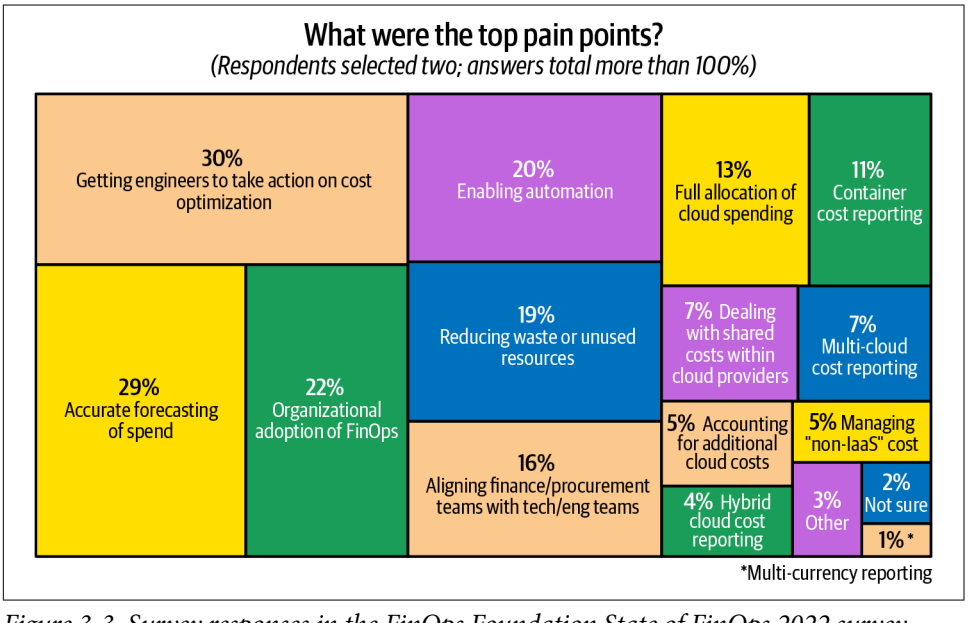


Figure 3-3. Survey responses in the FinOps Foundation State of FinOps 2022 survey, showing motivating engineers as the top challenge

Stories from the Cloud—Mike

I was once talking with a FinOps team inside a large tech company. The conversation moved on to motivating engineers to take action when one of the team members shared a story with me. His father had been working in tech for a few decades in Silicon Valley, and he was telling his dad about the challenges he was facing at work to get engineers to make changes to their cloud configurations. During this conversation, his father laughed and said something like: "You don't have a new cloud problem. You have an age-old problem of motivating other humans to do something you want."

It is easy to fall into the trap of thinking this problem is unique to the cloud. As a FinOps practitioner, you identify opportunities for optimizations and report the opportunities to your engineering teams. Recommendations require your engineers to spend time reviewing them and then making the changes necessary to their cloud resources. These individual recommendations have specifics on how various cloud resources are charged and configured and alternate options available. You can see how cloud-specific this problem can appear at an individual level and how new this practice of highlighting opportunities to engineers to follow up is.

It's important to step back from the cloud-specific concepts and begin to focus on the core issue, which is: how do you motivate others to make changes and incorporate new things into their workflow? Consider the task you are trying to achieve when you generalize the challenge to be as simple as "you need engineers to dedicate some time to perform a recommended action."

As seen in Figure 3-4, the action scale visualizes the balance between how likely engineers will be to implement your recommendations. The scale has many factors that tip either in favor of action or against.

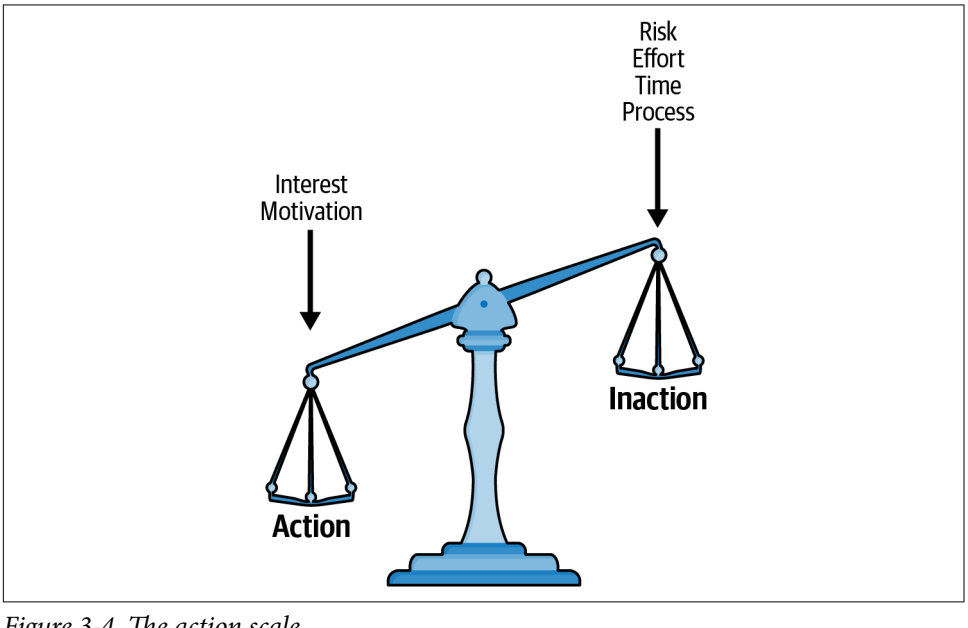


Figure 3-4. The action scale

Contributors to Action

On the side of the scale that tips it in favor of action, you have:

Interest
: How much interest do your engineers have in the recommendation? Instead of presenting a recommendation with "You need to review this recommendation," change the language to "Do you know what might be causing this recommendation?" Interest in the recommendation often leads to increased motivation.

Motivation
: The more your engineers have motivation—in terms of being measured or bonused on outcomes related to cost—to work on the recommendations made to them, the higher the likelihood that they will spend their time reviewing and implementing them.

Understanding
: For your engineers to take action, they need to understand the recommendations, how they have been made, how they should review them, and what is required to complete the changes necessary.

Detractors from Action

Opposing the contributors to action, you have the following detractors working against action from your engineering teams:

Effort and time
: We doubt that your engineers have plenty of free time. The amount of time your engineers need to spend implementing a recommended change detracts from the likelihood of taking action.

Process
: Motivation from your engineering teams will reduce as the amount of process that needs to be followed increases, especially manual steps and approvals.

Risk
: Recommendations don't come with zero risk. Changing the configuration of cloud resources can lead to unexpected outcomes. Trust in the recommendations is impacted once your engineering teams experience an issue or hear of others' failures.

Tipping the Scales in Your Favor

You can approach the challenge of driving action in two ways: drive up the contributors or combat the detractors. In reality, you will need to do a little of both.

Increase the motivation
: Engage with your engineers and ask them what it will take for them to get involved. Guide the way they approach the effort. However, try not to be too prescriptive and allow them the freedom to decide how the work will get done. When you have teams engaging in the process, show the success of their efforts to all engineering teams to inspire others, and finally, show appreciation and reward their efforts. We have seen teams use internal blogs or talks at town hall/all-hands meetings celebrating success as an excellent method of sharing the word. Offering stickers and T-shirts for completing training can be a great way to incentivize teams to learn about your program.

Educate teams
: Think through how you explain what you need engineers to do. Ensure your engineers understand why you need them to perform the task and within what timeframe you expect them to do it. If engineers are unclear about how they complete the task or its benefits, they will ignore the ask. Provide training, links to documentation on how the recommendations have been made and how to adjust the settings on the cloud resources, and make yourself available to answer questions.

Reduce the effort
: The more time you spend validating the quality recommendations, the less time engineers need to review them. Work with your engineers to determine what information you can provide alongside your ask. Bringing the relevant information together for your engineers to use all at once reduces gathering effort. Automation can reduce the time and effort it takes to implement the tasks.

Avoid a loss of trust
: After a failure, you need to understand how the task did not succeed. Remove high-risk recommendations and share with your engineering teams how they can better assess the recommendations to avoid a repeat incident.

Focus on building a company culture that enables finance to develop a partnership with your engineering teams and allows them to understand each other's motivations and develop a collective understanding of your organizational goals. To influence change, empower teams to get involved in the creation of the processes that need to be followed and identify items that grow people's interest in helping.

Conclusion

Throughout this chapter, we've emphasized that every team must adopt a FinOps mindset. All teams inside your organization are able to work together to understand one another's goals alongside a centralized FinOps team that is helping to build out reporting and practices to assist everyone in achieving them.

To summarize:

- All teams have a role to play in FinOps.
- Teams have different motivators that drive spend and savings.
- Teams need to work together with a balance of empathy for one another's goals.
- FinOps practitioners help align teams to organizational goals.
- Motivating engineers to take action is not a new cloud problem but an age-old challenge humans have faced.
- The action scale helps visualize the items that contribute to getting engineers to take action and the items that detract from it.
- To tip the action scale in favor of action, you need to implement programs to help the contributing factors and reduce the detractors.

In the next chapter, we'll look into how teams talk to each other. You'll discover that good collaboration requires more than placing teams in the same room. The language of finance can be vastly different from that of engineering, so everyone must embrace a common language.

CHAPTER 4
The Language of FinOps

Each team within your organization likely uses discipline-specific terms, views cloud billing data from a different perspective, and has separate motivators for what they would like to achieve. This chapter discusses how you can enable teams to collaborate more effectively by educating staff on specific terms being used by each and by defining a common lexicon to encourage alignment and trust.

Stories from the Cloud—Mike

Prior to starting a FinOps practice at Atlassian, teams tracked their own costs and used their own methods for deciding which parts of which bills were their responsibility. My first action was to create reports and send them out to the business teams. But I left out a vital step. Teams read the reports from their own perspective, just as they'd always done. My reports didn't clarify cloud spend—they confused it. The terms I used didn't make sense to everyone, so different teams ended up with divergent opinions about the reports' validity.

Seeing all of this, I realized we needed a common language. I needed to create a broader understanding of FinOps terms and a common set of reports that all teams could use to track their spend and optimizations. My previous reports, filled with cloud-specific infrastructure language, caused confusion and frustration for the finance team, just as the reports focused on cloud financials did for the engineering teams.

An early stage company typically starts with simple daily spend visibility that shows teams their respective spend. However, the common lexicon is important not only for those starting out in FinOps, but even the most well-versed FinOps practices. As your maturity in the cloud increases, the amount of terminology used in your cost reporting expands. As you continue to add capability areas in your FinOps practice,

increase consumption of additional cloud services, and become more granular in your reporting, you end up with even more discipline-specific terminology. This increases the potential to use ambiguous (or duplicate) terms, which can create confusion and lack of trust in the data. Consistency becomes even more important in later stages to provide a better understanding of how all the reports relate to each other and drive accountability to the respective teams. Try to avoid using esoteric discipline-specific terms from finance or engineering and aim to agree upon a standardized lexicon.

Defining a Common Lexicon

It's easy to highlight the need for a common lexicon: simply ask a finance team and an engineering team to each describe a service. A finance team typically thinks about the costs and rates of services while engineers will be thinking more about their *service availability, reliability*, and *available capacity*. Both are correct. The disconnect comes when these teams try to communicate with each other, which cloud forces them to do much more often than was common during the data center days. In that not-so-distant past, operations teams interacted with costs only when they proposed new equipment purchases. After that, their day-to-day job was about the efficiency and performance of their services. Finance teams, alternately, focused on what spend was committed and which budgets it should be depreciated against.

When you distribute technology buying power throughout your organization, you need to ensure all teams are using the same vocabulary for describing costs—and that you don't overload terms to mean different things to different people. The language gets more complex when you add in cloud service provider–specific terms, and it quickly approaches incoherence as you switch between multiple cloud service providers and their vendor-specific nomenclature.

If every report you generate also needs to come with a dictionary of terms—or worse, someone has to sit down and teach a team how to read a report—that prevents the interteam collaboration at the heart of FinOps. People will refuse to give reports the time it takes to understand them, while the FinOps practitioner quickly will run out of bandwidth to keep everyone informed.

To build a common language across an organization, reports must consistently use specific terms, and teams must have learned how to correctly interpret them. While one person might understand that *divided costs* means the same as *cost allocation*, that won't be common knowledge in the beginning of FinOps.

Where possible, it's better to use existing language constructs instead of creating all-new terms that teams must learn. And if a cloud service provider uses terminology that doesn't align with existing business language, it's best to translate it before reporting out to the teams.

Defining the Basic Terms

Of course, having everyone read this book will also help build a common understanding of the terms used in FinOps. Let's define some terms used in the industry:

Cost allocation (or cost attribution)
: The process of splitting up a cloud bill and associating the costs to each cost center. Chapter 12 covers this process more thoroughly. It's important to have teams understand how costs are being allocated, and to have a centralized, controlled, and consistent cost allocation strategy.

Wasted usage
: Resource usage that isn't actively used by an organization. If a resource is provisioned, a cloud service provider will still charge for it—even if it isn't used.

Oversized/undersized
: When a cloud resource is provisioned larger than is required, such as having too much memory or compute power, it's considered oversized. It is worth noting that the opposite can be true as well. For undersized resources, you can increase the size if it derives more business value from additional power.

Rightsizing
: Rightsizing is the act of changing the size of provisioned resources to one that better matches needs.

Workload management
: The practice of ensuring that resources running in the cloud are only running at times when they are needed. It is one of the practices that allows engineers to optimize usage.

On-demand rate
: The normal or base rate paid for a cloud resource. This is the public pricing for a cloud resource.

Rate reduction
: Using commitment-based discounts such as Savings Plans (SPs), Reserved Instances (RIs), Committed Use Discounts (CUDs), Flexible CUDs, or commercial agreements between an organization and a cloud service provider to receive a lower rate for the resources used.

Cost avoidance
: By reducing resource usage, either by removing a resource altogether or by rightsizing it, you can avoid paying for resources that would have incurred a charge. This method of reducing cloud spend will be covered in Chapter 15. Note that there will be nothing in billing data that actually tracks cost avoidance; it's

often measured as a reduction in the amount of cost for the current month's billing cycle.

Cost savings
: By reducing the rate you pay for resources, you generate savings. Unlike usage reduction where you avoid costs, cost savings are represented in your billing data. The usage is there, but you pay a lower rate for it. Usually you can track savings in billing data, either directly by monitoring credits applied to a bill or by comparing the rate paid for a resource versus the normal public price. Engineers should confirm with their finance departments the meaning of any use of the word "savings," however, as some finance departments interpret savings as the ability to reduce budget.

Savings realized
: When a saving is applied to billing data, you're able to track the amount of savings you've generated in your cloud bill. By tracking realized savings against the efforts to generate and maintain them, you're able to determine the overall effectiveness of your FinOps practices.

Savings potential
: When looking at your cloud bill forecasts, you can predict the amount of savings using your existing commitments and commercial agreements. But until these savings are applied to your accounts, this is only savings potential.

Commitment-based discounts
: By precommitting to a cloud service provider a set amount of resource usage using SPs, RIs, or CUDs, you receive a reduction in the rate normally paid for those resources.

Commitments unused/unutilized
: For every hour you've committed to a resource usage that you don't use, that reservation goes unused, or unutilized. Another term for this is *reservation vacancy*.

Commitment waste
: Having a reservation with some amount of underutilization isn't an issue as long as the discount you're receiving is larger than the cost of the unused reservation. When the reservation is costing you more than what you would save—that is, it's not utilized to an amount that saves you more than the cost of the reservation—you call this commitment waste.

Covered usage
: When a resource charge is discounted by a reservation, you call it covered. The usage is being covered by the reservation, and the result is a lower rate of charge.

Coverable usage

Not all usage in the cloud is coverable with a commitment. For example, if you have resource usage that spikes during business hours and then is removed after business hours, making a commitment might result in commitment wastage and wouldn't save money. However, it's also possible that some overnight batch work starts as the day usage rolls off and a commitment would cover both workloads, resulting in savings for both. When covering usage with a commitment would result in savings, classify it as coverable. What is important is for a set of resources to provide enough consistent usage for a commitment to be utilized to save money, not whether a single resource runs consistently.

Unblended rates (AWS specific)

Some resources are charged in decreasing rates the more you use them. (See Chapter 16 for more on volume discounts.) This means you're billed different rates for resources as you use more, or for longer periods during the month. By examining your bill, you can see that some resource costs are larger than others, even for the same type of resource or an identical resource. When the rates are presented this way, they're called unblended in the AWS vernacular.

Blended rates (AWS specific)

AWS offers a blended rate in their billing data. This blended rate standardizes the rate you pay for the same type of resource by evenly distributing the charges to each resource, except when the usage is covered by commitment-based discounts. While AWS offers this blended rate in their detailed billing data, the way the costs are blended is often not perfectly even, or some resource costs are not blended because they are covered by commitments, which can lead to confusion about the true cost of a resource. There is still a specific use case for blended rates in reconciling invoices when using Cost and Usage Report (CUR) data files, but beyond this blended rates are rarely useful.

Amortized costs

Some cloud resources and commitment-based discounts can be purchased with an up-front fee. The amortized cost of a resource takes this initial payment into account and divides it out, attributing the prorated cost for each hour (or second, or millisecond) of billing. For example, a one-year all up-front commitment vehicle (SPs, RIs, or Reservations) will be split up over all the periods of time where that commitment was applied to a resource.

Fully loaded costs

Fully loaded costs are amortized, reflect the actual discounted rates a company is paying for cloud resources, equitably factor in shared costs, and are mapped to the business's organizational structure. In essence, they show the actual costs of your cloud and what's driving it. Your organization may choose to define fully loaded costs differently based on the discounts and cost components you wish to

include. Or you may define another term to represent this in your organization's reporting.

Defining Finance Terms for Cloud Professionals

Some terms that come from the finance discipline that are helpful for the engineers to understand include:

Matching principle
: Expenses should be recorded in the period in which the value was received, not necessarily during the period the cloud provider invoiced them or when payment was made. The matching principle applies to the accrual basis of accounting and is the main difference from the cash basis of accounting. In IaaS billing, the (typically monthly) invoice you receive from the cloud provider may include up-front payments that will apply benefits—discounts—to resources over the longer (12- or 36-month) term of the commitment; most of the benefit of the up-front payment is not in this accounting period. This means you should expense spending using the billing data (e.g., Cost and Usage Reports in AWS, Azure Billing File in Azure) rather than using the invoices from the provider.

Cost of capital/WACC/ICC
: Cost of capital refers to the cost to an enterprise to deploy their money toward an investment. Sometimes this is referred to as the internal cost of capital (ICC). In cloud, cost of capital is an important consideration when looking at paying up front for additional discounts on commitments like RIs. For example, if a company borrows money at 8%, then its cost of capital is 8%. That 8% becomes the bar the company needs to exceed in its return on investment. However, this 8% example is vastly simplified. In reality, most companies have a variety of sources through which they gain access to capital. Various types of debt and equity financing may bring very different rates. When doing cost of capital calculations in such a situation, companies must use a blending of those rates, called the *weighted average cost of capital* (WACC). Most finance teams will know what their WACC is and must consider it when making up-front RI purchases.

Cost of goods sold (COGS)
: COGS measures how many dollars of outlay it takes to generate revenues in a specific period. For example, if a power company is trucking coal out of storage and into the power plant, it would record the cost of the coal burned. That cost has no future benefit, so it's going to be an expense that is directly traceable to revenue in that period, making it a COGS expense. The test of COGS is: are they directly expensed and directly related to revenues in the same period?

Hosting costs spent on R&D may receive more favorable tax treatment than that spent on COGS, so the ability to clearly distinguish between the two is a potentially valuable capability of the FinOps effort.

—Jason Rhoades, Intuit

For a software company, the COGS would be the monthly cloud bill to operate its software, salesperson commissions, and support costs. Notably, cloud is the most variable and has the most potential for optimization. Since it isn't generally good business practice to turn down your sales commissions or fire your support people, this shines a bright spotlight on optimizing your cloud spend without reducing revenue.

When COGS Can Become Capitalized Assets

There's a potential curveball when it comes to how expenses are used. Let's say a power company takes some of its coal and uses it to make diamonds. If it burned coal to generate power that was sold for revenue, the company accounts for the cost of the coal as COGS. But if it creates diamonds out of the coal, and those diamonds aren't sold in the period but instead are put into storage as inventory to be sold in some future period, the cost of the coal would be R&D spend and therefore capitalized as an asset. However, as soon as those diamonds are sold, then the cost of the coal switches back to COGS during the period of the sale.

To illustrate, we previously saw a retailer that was developing a new shopping platform product and was applying COGS in an interesting way.

The company accounted for the cloud compute hours used during the generation of the product as a capitalized asset that wasn't expensed in the period. It was able to do this because the product in development wasn't generating any revenue. The retailer was using cloud compute hours to create an asset that would generate revenue in future periods, much like the power company creating diamonds from coal rather than burning it for power.

Once that shopping product went live, the capitalized compute costs began to be amortized into the relevant periods in which the product began to generate revenue. Note that the shopping platform product is not a physical asset, so it was amortized and not depreciated.

In cloud, it's common for RIs to be amortized into the period in which they are used over a one- or three-year period.

EBITDA

Earnings before interest, taxes, depreciation, and amortization (EBITDA) is a widely used metric of corporate profitability. This metric also excludes expenses associated with debt by adding back interest expense and taxes to earnings.

Nonetheless, it is a more precise measure of corporate performance since it is able to show earnings before the influence of accounting and financial deductions. EBITDA can be used to compare companies against each other and industry averages. EBITDA is a good measure of core profit trends because it eliminates some extraneous factors and provides a more accurate comparison between companies.

Abstraction Assists Understanding

Human beings tend to have trouble with very large and very small numbers. For instance, if something costs $1 per hour, calculating how many hours you would get for $10 is relatively simple. However, if a resource costs $0.0034 per hour, working out how many hours you would get for $10 will require a calculator. Number formatting is also important. Without formatting, 100000000 is hard to read. Most people would need to count the zeros and add commas to determine the amount it signifies.

Another human issue with large numbers is that we tend to lose our sense of scale. This can be a problem with teams working with large dollar amounts, like thousands, millions, tens of millions, or more. As Hans Rosling points out in his book *Factfulness* (Flatiron Books), the size instinct kicks in when humans deal with very large or very small numbers, and you need to identify ways to deal with this.

In a FinOps Foundation meeting, one of the members showed how using abstraction can assist FinOps. He strengthened his point by noting that the difference between one billion and two billion doesn't sound like much, but it's actually a huge change.

Many organizations with large cloud spend struggle to give their engineers meaningful metrics about spending. Imagine an organization is going to spend $100 million a year on cloud. At that scale, all humans will struggle to understand the scale of the numbers involved. And when you have trouble with scale, it becomes difficult to answer questions like these:

- Is a $30,000 a month optimization a little or a lot?
- How much effort is worth putting in to achieve the savings?
- Does it really matter to the business?
- Are there more important priorities?

Stories from the Cloud—J.R.

J.R. worked with a Fortune 50 cloud spender who refused to action the $1.5 million per month of commitment-based discounts available to them for more than a year. The constant refrain from the VP of Technology (who was also the COO of the IT portfolio) was "I have a two billion dollar a year IT budget. Those optimizations don't move the needle for me given the effort required to action them." He had bigger fish to fry, which was the more pressing corporate goal to shut down data centers as quickly as possible. Though it may not look like it, the VP was still doing FinOps by practicing the *informed ignoring* concept covered in Chapter 2. He knew about the optimization available to the organization and the effort required to achieve it; once this value versus effort equation hit a threshold for the organization, action would be taken. Had the organization prioritized performing the optimizations instead of focusing on shutting down their data centers, the savings made would have been negligible in comparison to the added costs for having to negotiate contract extensions with their data center provider.

Keeping a culture of accountability, so vital to a successful FinOps practice, becomes more and more difficult when the numbers get too large to relate to.

To help with context and understanding, it's sometimes helpful to not use the common lexicon, because the specific number of dollars spent or saved may not be the key takeaway. If the impact to the business for spending or savings can be articulated using other units of measurement, the message comes across much more clearly.

For example, a FinOps Foundation member correlated the amount of cost savings to beer. If the team took a certain set of actions, he showed that they could save the equivalent of a thousand beers. In the years since the first edition of this book, that same *beer metric* practitioner has been able to evolve his reporting to include tangible business metrics such as requests served, revenue delivered, and customers served to their offering.

This move away from reporting only in dollars is important, because each team involved with cloud spend has a different set of motivators. Using the motivations of teams to find an appropriate example is one way to put cloud spend in a more meaningful context. For example, an up-and-coming motivator of engineering teams is the introduction of sustainability metrics like carbon emissions. See Chapter 19 for more on sustainability and FinOps.

For business leaders, seeing cloud costs tied to an important business value metric, such as a percentage of the supported product's revenue or the output of some key activity, gives a more understandable picture of the overall efficiency of your cloud spend. An organization can then set target limits for the spend percentage and use

this to drive projects to optimize using a number of techniques to reduce growth of cloud spend relative to revenue growth. Part III discusses optimization methods.

But that overall business strategy doesn't directly relate to teams' motivations. Engineering teams, for example, turn their efforts into functionality for users, so their motivations revolve around growing their engineering team and delivering more functionality. A FinOps Foundation member found that by reporting cloud spend in terms of the monthly cost of the engineering team, he had landed on a meaningful way to engage the engineers. When evaluating a cost-saving action, they were able to say that an optimization action might result in a specific amount of additional headcount, or they could say how quickly a cost-saving action might pay for a team's salary.

Knowing the teams' motivations helps you find an effective way to engage them. For service managers, reporting in alternative units like cost per daily active users (DAUs) is a good tactic. By dividing the cloud spend by the number of DAUs of their services, they can show the amount of business value those services are generating per dollar spent.

As a FinOps practice continues to mature and you start adopting unit metrics, you can associate a true cost to a business function and start to steer *cost* conversations to *value* conversations. For a transport company, your unit metric might be packages delivered, or you might start with a simpler metric like cost per request or API call. When you can equate a cloud cost to each business delivery, you can report on cost optimizations in terms of how many more shipments you can make from the savings you generate. All of this contextualizing helps create a common understanding among teams.

Cloud Language Versus Business Language

As you create reports that move away from the people operating the cloud resources, or from the FinOps team that's operating the cost-saving programs, it makes sense to abstract away the cloud-specific terms. When you switch those terms for more generic business terms, you can summarize reports while still being accurate.

As an example, let's look at cloud reservations for compute resources. These will be covered in more detail in Chapter 17, but generally the idea is that if you make a commitment to a cloud service provider and use those commitments effectively, you can save money. If you don't, you lose money. AWS calls them Savings Plans or Reserved Instances, with various options saving different amounts.

If your FinOps team creates a report for themselves, they need the details on the individual commitments made. But as you start reporting up to finance and then to the business at large, the important message changes. What you focus on here is the overall success: did you save money or lose money? When you move from

reporting with cloud-specific terms like *commitment utilization* into more generic business terms like *savings realized*, you create clarity.

Reducing the number of individuals that need deep cloud knowledge, especially as you move farther away from the technology toward more general business reporting and monitoring, reduces the knowledge barrier. And focusing your reports on the actual business outcome being reported continues to foster more trust, credibility, and understanding.

Creating a Universal Translator Between Your DevOps and Finance Teams

We previously referred to this using the concept of a *Babel fish* from the Douglas Adams masterpiece *The Hitchhiker's Guide to the Galaxy* (Harmony Books). However, it was a reference that not everyone understood. Instead we are going to use a *Star Trek* reference to a similar idea called a Universal Translator. In *Star Trek*, the Universal Translator is used to interpret alien languages into the user's native language. It's exactly the kind of thing every FinOps team wishes they could give to finance, technical, and product teams so they could all perfectly understand one another.

For example, we've been in meetings where the technical team talked about the specific values they use to tag cloud resources while the finance team talked more broadly about the need for cost allocation. Both teams were describing the same thing, but there was nuance to the reporting they were reviewing, such as details of how a tag value was used to allocate costs, that prevented these two teams from understanding each other.

They could have used a translator to cut through the confusion.

But if both the finance and technical teams are clear on how FinOps practices such as cost allocation work within the organization, the conversation always starts from a common level of understanding—no universal translator needed.

Finance and engineering teams are very smart. However, you must remember that they have different practices and different terminology. FinOps should help finance teams to understand the cloud language by abstracting the cloud-specific terminology away for those who understand the dollars and cents, and they should simplify the finance requirements for the engineering teams.

FinOps practitioners build reporting and processes that reduce the need for both finance and engineering teams to spend large amounts of time learning and working outside their areas of expertise. Building those reports with consistent language enables teams to familiarize themselves with a common set of terms and then use those common reports to have conversations around the cloud benefits and costs.

You need FinOps to enable teams to communicate efficiently. Ideally, someone from the FinOps team isn't required in the room at all. That person's presence becomes less necessary when everyone shares a vocabulary. However, the FinOps team is always there to assist in building these reports, so that trust, understanding, and clarity will continue to grow.

The Need to Educate All the Disciplines

FinOps teams can then help collect and define the terms used by an organization that will be used to describe cloud spend and savings. Discipline-specific terms, industry terms, and variations used by different cloud service providers must be adopted, and/or adapted, into a common lexicon.

No one in finance, technology, product, procurement, or management should be expected to try to learn all the common languages of every other team. Ideally, everyone meets in the middle, where finance learns the necessary terms used to describe cloud resources, engineering learns the terms used to describe costs, and product teams learn to communicate with both. When teams learn more of the language used by other teams, the conversations lead more quickly to successful outcomes.

Benchmarking and Gamification

When common reporting built around the same language is used to measure each team's spend and optimization, it becomes possible to compare the teams—and even create some friendly rivalry. Chapter 22 digs deeper into the metrics used to compare groups, but for now think of how being able to compare teams can lead into *gamification*.

For example, virtual achievement badges can be awarded for successful management of team cloud spend. Celebrating teams that perform the most optimizations, and highlighting the specific positive effect on the overall cloud spend and optimization, is a great way to engage and encourage teams.

Alternatively, having reporting that highlights the worst offenders—those who have shown a lack of optimization actions, ignored their team's cloud spend, or were slow to respond to unexpected spend anomalies—can be effective and even fun, if done well. From our experience, teams don't like to be listed as the worst offenders and will often put in extra effort to change their position on the list. We will look more at what we call "shameback/worst offender lists" in Chapter 11, when we get to practices that will help you reach your FinOps goals.

Conclusion

Ultimately, you use your common lexicon to build a shared understanding of your cloud costs and optimization opportunities.

To summarize:

- Be aware that different teams use discipline-specific terms.
- Help staff learn common vocabulary and stay consistent with the terms used in reporting, which will help eliminate confusion.
- A FinOps team doesn't have to be a constant translator in meetings but should assist in learning and enabling teams to communicate more and more on their own.
- Consider adding reporting using abstracted units of measurements to communicate in ways that are more meaningful to your teams.
- Dividing out costs and savings against some unit of business value allows you to gauge how efficient your cloud spend is.

Before you can optimize your cloud spend, you need to understand your cloud costs. We'll take a look at the anatomy of a cloud bill in the next chapter.

CHAPTER 5

Anatomy of the Cloud Bill

In this chapter, we'll examine the fundamentals of how public cloud services are billed and the way those charges are represented in billing data. Understanding a cloud bill is key to being able to allocate and optimize cloud costs later in the FinOps lifecycle. The structure of an organization's cloud spending will also help determine who will perform certain optimization actions, as well as how FinOps work will be delegated.

Instead of looking at individual billing lines (which is sometimes required in FinOps data deep dives), this chapter looks into how cloud service providers present data to companies to charge for cloud resources. Correctly understanding this will help FinOps practitioners to create impactful reporting that will assist all teams in making sense of the cloud bill. Chapter 8 uses these concepts to influence the user interface (UI) of FinOps.

Types of Cloud Bill

The top cloud service providers offer you access to your cloud billing data in one of three ways:

An invoice
: This is lousy for FinOps, as it's very summarized and does not provide enough data granularity or freshness. Primarily, invoices are useful for finance and general accounting purposes.

Cloud native cost tools
: Native tools (e.g., AWS's Cost Explorer) visualize your data and can be great if you're using only a single cloud service provider and if your reporting needs are fairly basic. But as the complexity and scale of your cloud environment

grow—especially for multicloud organizations—native tools have limitations that can impact your FinOps maturity.

Detailed cost and usage billing data
: Detailed billing data (e.g., AWS's Cost and Usage Report or Google Cloud's BigQuery Export) is usually accessible via file(s) or API. This data is great for seeing all the details available, but it's complex to use and requires specialized knowledge and data analysis tooling to understand.

Cloud Billing Complexity

Cloud billing data is complex, and it must be to enable FinOps. In the first edition of this book, published in 2020, we stated that AWS alone has over 200,000 individual product SKUs, some that are billed at per-second resolution. As of the second edition, three years later, that number had grown to over 791,000 individual SKUs listed in the AWS pricing API alone, not counting those offered by Azure, Google Cloud, Oracle, and others. Detailed billing data typically comes in through multiple updates each day, with a complex interconnection of charges behind spending, such as instance hours, gigabytes of storage, and data transfer. We've seen large cloud spenders with billions of individual charges each month in these files.

While there are platforms and cloud native tools (as covered in Chapter 8) to help decipher the complexity, it's important to have at least one FinOps practitioner on the team with a deep understanding of the data. Not only will this help others understand and unpack billing concepts, but it also makes it easier to interpret the data and recommendations coming out of FinOps tooling. Billing experts have been known to identify discrepancies in billing data and report them back to their cloud service providers for billing adjustments and/or refunds.

The invoice data a finance team looks at may not always appear to align to the detailed billing information (e.g., the AWS Cost and Usage Report) that the team may be analyzing. While an external FinOps platform can help align the two for invoice reconciliation each month, the levels of granularity with which invoices apply amortizations, blended rates (in the case of AWS), or commitment-based discounts (such as Savings Plans [SPs] or Reserved Instances [RIs]) can be different than the detailed billing data at the service or account/subscription/project level. Therefore, it's helpful to set an expectation early that the team should use invoices only for the purposes of accounts payable, not for analyzing portions of cloud spend.

Invoices largely are an accounts payable tool, not a FinOps one. FinOps requires much more granular and timely data than a rolled-up monthly bill can provide.

To identify cost drivers of any service, you need to get below the monthly detail and cloud service level provided by the invoices. The major cloud providers (such as AWS, GCP, Azure, and Oracle) all offer solid native cost tools to get you started analyzing spend at a high level, and those help us take that first important step into visibility. Native cost tools still have some limitations such as handling multiple cloud bills once an organization goes multicloud, handling custom negotiated rates, the need to push out allocated spending data to create team accountability, or container cost allocation. Keep in mind, though, that the cloud native tooling can also always be used to perform research and ad hoc reporting into specific costs, so you should develop an excellent understanding of these tools and how to use them effectively even if you rely on third-party or homegrown tooling.

Basic Format of Billing Data

Let's start with the basic format of most of the billing data that comes from each of the three major cloud providers. Each row in the file enumerates the usage quantity of a certain type of resource used. The attributes of a cloud billing item also tend to include a snapshot of usage during a specific time period. Attached to the usage row will be the following:

- Time period
- Amount of the resource used
- Rate details used for the charge during that period
- Where in the world it is located by region
- The resource ID
- Metadata attributes—like account, project, or tag—that can be used to allocate the spend to a product

Later, Chapter 12 covers tagging and how to utilize it to drive accountability. For now, let's look at some samples of billing data from each of the main cloud providers so you can start to understand the raw elements that fuel your FinOps programs.

Figure 5-1 shows a single line of billing data, one of potentially hundreds of millions of lines of billing data received daily. To the untrained eye, the data can appear illegible, but you can discern a number of important attributes:

- When the resource was used
- The rate and charge applied to the bill
- The resource being charged
- Whether a reservation was applied, and if so, which one

- Tagging information and metadata that helps with cost allocation
- The region and service of the charge

2019-07-22T00:00:00Z/2019-07-22T01:00:00Z,,AWS,Anniversary,01234567890,2019-07-01T00:00:00Z,2019-
08-01T00:00:00Z,09876543210,DiscountedUsage,2019-07-22T00:00:00Z,2019-07-
22T01:00:00Z,AmazonEC2,BoxUsage:m3.medium,RunInstances,us-east-1b,i-
0dd86597c590879b9,1.0000000000,USD,0.0000000000,0.0000000000,0.0402772184,0.0402772184,"Linux/UNI
X (Amazon VPC), m3.medium reserved instance applied",,Amazon Elastic Compute Cloud,,,,,,2.5
GHz,,,No,,,,,,,,,,3,,,,,,,,,,,,,,,,,,,,,General purpose,m3.medium,,,No License required,US East (N. Virginia),AWS
Region,,,,,,,,3.75 GiB,,,,,,,Moderate,Linux,RunInstances,,,Intel Xeon E5-2670 v2 (Ivy Bridge/Sandy Bridge),,,64-
bit,Intel AVX; Intel Turbo,Compute Instance,,,,,,,,,,,AmazonEC2,ASDZTDFMC5425T7P,,,,,1 x 4
SSD,,,,,,Shared,,,,,,,BoxUsage:m3.medium,1,,,,,,3yr,convertible,No
Upfront,0.0670000000,0.0670000000,Reserved,Hrs,,,arn:aws:ec2:us-east-1:01234567890:reserved-
instances/36768f14-be7f-455f-9657-6d1c4e06401a,,,Infrastructure Svcs-
Software,,mfuller,serviceA,,Infrastructure Svcs,Infrastructure Svcs,,,2.0,2.0000000000,,,,,,,,,,,,,,,,,,,,2,,,,us-east-
1,Amazon Elastic Compute
Cloud,,,m3,,,,serviceA,,0.0670000000,0.0670000000,0.0670000000,0.0670000000,,,,0.00000000,,0.04300000,,,0.
04300000,,,,,,,Public,"38.953116,-77.456539",,,,,,s3://billing-
reports/cost_reports/hourly_with_resources/20190701-20190801/2755e466-dc41-4a04-9d9e-
f1771805729b/hourly_with_resources-059.csv.gz,,"Amazon Web Services, Inc.",Used,,,,,,,,,,,,,,,,,awseb-e-
myky2mqvfv-stack-AWSEBAutoScalingGroup-WAUJKY5MFRGL,,14292253,1947602408,infrastructure
svcs,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,2019-07

Figure 5-1. Sample line from the AWS CUR billing data

All this granularity gives a FinOps team incredible power as the practice is built out. But granularity creates complexity, and very quickly billing data gets so large that it can't be managed with spreadsheets. Any material amount of cloud spend will generate a massive volume of charges: a sizable cloud spender may have billions of individual charge line items per month with hundreds of columns of meta detail attached to each. The more cloud native you are, the bigger the data set: serverless functions, ephemeral instance use, per-second billing data, and so forth are driving a big data problem for the FinOps practitioner. You must turn to big data analysis techniques for help.

However, you cannot simply throw a data analyst at the raw billing data. None of the existing financial systems used for managing traditional IT spend can adequately handle the volume of data without summarization, nor do they provide data freshness at the near-real-time speed required for FinOps to create a feedback loop.

Each cloud has different levels of granularity and detail, as well as different terms for their dimensions and metrics. Each of the various commitment-based discounts are represented differently in the way they are applied. A skill set is required on the FinOps team to fill the data engineering role (as described in Chapter 3).

Handling the data should be a key consideration when deciding between building in-house tooling versus buying a vendor platform, which is discussed in Chapter 8. The vendors abstract away the problem of keeping your data up to date as you add

cloud and services, as well as the data quality controls related. The clouds themselves also frequently adjust their own data when new services are added, or new billing functions are changed. Managing the data flow and those changes is a huge part of managing FinOps tooling.

Understanding this complexity allows you to do really cool things down the road, like automated anomaly detection when something *small* changes. We say small because cloud efficiency often suffers a death by a thousand cuts. Imagine your company spends $1,000,000 a month on cloud, and then a team leaves on an unused cluster of 50 instances at a cost of $5,000/day. Given the large difference in scale between those two numbers, you likely wouldn't even notice the change when you review service-level summaries or monthly roll-ups, and yet that *small* amount adds up quickly to over $150,000/month—or 15% of the total monthly spend. You need to be running machine learning on the data to analyze small changes before they add up to big costs. Luckily, the cloud providers give you incredible power through the granularity of data to address small cuts before they turn into the infected wounds of overspending.

A Simple Formula for Spending

The formula for a cloud bill is really simple:

$$\text{Spend} = \text{Usage} \times \text{Rate}$$

Usage might be the number of hours (or seconds) a resource is used. The rate is the hourly (or per second) amount paid for the usage of that resource, or the class of storage used. Conceptually, it's pretty simple. Change either of those items and your cloud bill goes up. Increase both and it can go way, way up.

Time, Why Do You Punish Me?

In the '90s, the band Hootie and the Blowfish released a song called "Time." They sang of how time can "teach you 'bout tomorrow and all your pain and sorrow." And that "tomorrow's just another day...and I don't believe in time."

Hootie is unlikely to be effective at FinOps because cloud billing is all about time. Nearly every charge is based on time. Even items that are flat rate are charged over the period of a month or a year. More commonly, you're charged for a second of compute, a *GB month* of storage, or however long it took for a query to complete. This all relates to the *usage* component of the formula for spending: how much of a cloud resource you use over a period of time. There are, of course, a few exceptions

to this rule. Serverless and data transfer are not time-based but rather volume-based. Both still fall under the same model of *volume* × *rate*.

For example, 720 GB of data transfer could be charged at a rate of $0.02 per GB. Or 235,000 function requests are charged at $0.001 per request. However, these examples are still based on usage. Did you use the thing? If yes, then you're charged. If no, then you're not charged. It's another example of the variable spend model of cloud versus the fixed spend model of on-premises.

An important cloud billing nuance to remember: you're charged if the resource is on, regardless of whether the resource was used efficiently, or not used at all. You're solely responsible for ensuring effective usage and appropriate sizing of resources.

Consider how a compute resource is charged by the cloud provider. The charge is based on how long the resource ran. Although most of the major providers charge for compute by the second, let's simplify this example by using hours. There are 720 hours in a 30-day month, and 744 hours in a 31-day month (we'll talk about February later in this chapter).

So if you were to look at a compute charge for a resource that ran during all of January and is billed on an hourly basis, you'd see a line that showed 744 hours of compute usage for that resource. Of course, it would also include a rate that was billed for those 744 hours. It's possible that all 744 hours could be billed at the same rate, but it's equally possible that, for example, 200 of them had an SP or RI applied, while the other 544 did not. If so, there would be two rows of data. One would list the 544 hours at on-demand rate, and the other would list the 200 hours under the SP/RI rate. Same resource, different costs at different times due to seemingly random application of commitment-based discounts by the cloud provider in the billing data. This scenario is the norm and makes normalization—and explanation to the business—of fluid cloud billing rates a constant headache of the FinOps provider.

Sum of the Tiny Parts

All of these tiny charges are aggregated to become the total charge per service or per month. But each tiny charge has its own nuances and attributes that, if examined more closely, can give you rich, useful data about your usage. Remember, you're being charged for the actual time in the period that the *thing* was *on*. Not whether it was *used*, but whether it was *on*.

We often explain it this way: "Don't get attached to resources. You're not buying discrete resources—you're buying a proportion of usage of a resource over time." In J.R.'s first job in IT back in the mid '90s—at the W. M. Keck Observatory in Hawaii—

he worked with servers each named after a local beach: *Hapuna, Kiholo, Kaunaoa, Maumae*. But in the cloud, compute resources should be thought of as ephemeral and replaceable. They are cattle, not pets.

As we'll explain further later on, even commitment-based discount programs like SPs/RIs/CUDs don't pay attention to what server they're attached to. They simply attach to one that matches their specific relevant attributes or provides the most savings opportunity. When you apply the same mindset to cloud billing, you realize that *you're buying time, not things*. And even though it might seem obvious to some, this concept is key for finance teams to understand as they begin to account for and understand cloud bills.

A Brief History of Cloud Billing Data

Full disclosure: we're complete nerds when it comes to cloud spend data. One might find us at AWS re:Invent or Google Next, wistfully reminiscing over the various iterations of cloud billing files over the last decade and the capabilities each iteration unlocked. At the risk of going down a nerdy rabbit hole, those iterations actually map perfectly to the typical journey of a company starting FinOps. Each step of the iterations reflected the maturity of the most advanced cloud spenders of the era. If companies weren't along for the ride during that time, they're quite likely currently on a similar journey. So, from a gradual adoption perspective, it's useful to quickly take a deep dive into the past 10+ years of AWS billing file updates. We're using AWS to describe the history as they were the first cloud provider to supply this type of billing data.

2008: Invoices
: This is where it all began. Here you saw what you were getting billed at the monthly level and service level, with no company metadata such as tags and no granularity to understand change or cost drivers. A finance team would look at this and invariably ask, "Is this invoice right?" To answer that question, you'd need to look to the next iteration of the billing data.

2010: Consolidated Billing
: It may seem crazy now, but there was a multiyear period where AWS did not allow you to roll up payment to a single account. Early pioneers on cloud spend sometimes had dozens of disparate cloud accounts with separate bills that could not be paid or consolidated under a single one. Cloud sprawl became a serious problem: early cloud adopters often had rogue cloud spend accounts still being paid for on individual credit cards that had to be tracked down via expense reports. Consolidating billing added the concept of a *payer* and *linked* account hierarchy, enabling a synthetic billing relationship independent of security or permissions. This changed the game, not only for reporting purposes, but also making it easier to determine commitment levels.

2012: CAR (Cost Allocation Report)

Back in 2012, AWS users wanted to start answering questions like, "OK, I know I'm spending $100,000 this month, but which of my products or teams drove that spend?" The CAR file introduced the concept of individual rows of data for each tag value and each linked account. At the time this was a big deal—you could finally allocate spend. But it was also frustrating because it reported spending at a monthly granularity, so you couldn't determine when the spend started or when any spikes occurred. In the CAR there were also monthly aggregate lines for untagged spend, but there was no ability to see what resources made them up.

2013: DBR (Detailed Billing Report)

The DBR introduced time series to spend by service numbers. Instead of just being able to see which service cost how much in a particular month, you saw when in the month the service incurred that cost. This let you start to understand elasticity and intramonth changes made by teams that ended up impacting spending. But—spoiler alert—the DBR lacked a key piece of data that was about to change the world of cost management.

2014: DBR-RT (Detailed Billing Report with Resources and Tags)

The DBR-RT was a game changer. It introduced a single row for every resource used in every time period, with a column for each and every tag key. The data increase was staggering: a large customer's DBR rarely exceeded megabytes, but a DBR-RT could run into hundreds of gigabytes of CSV (comma-separated values) data. For some large spenders, that could mean billions of commas in a single file. This file let you see which specific resource ID in which specific time period with which specific tag was driving spending, and whether there was an RI applied *in that hour*. Suddenly, you could pinpoint changes with deep precision. As a result, early FinOps practitioners could start to give much better recommendations for optimization.

2017: CUR (Cost and Usage Report)

Coden-amed "Origami" during development, the CUR file was a complete rethinking of the billing schema. For the first time, AWS moved away from CSV into a JavaScript Object Notation (JSON) format that was better for programmatic ingestion. As part of this evolution, certain data, like rates, was split into separate JSON files, making it trickier to build your own billing data readers, which many people had done for the simpler DBR format. That's OK, though, because CUR had a small but powerful advantage: it could tell you not only *whether* an RI was applied but also *which* RI (or part thereof) was applied in that hour. Suddenly, you had a much clearer understanding of the utilization of RIs, and of how they were informing additional purchasing and modification decisions. In the years since 2017, AWS has continued to iterate on this file format, and it was still the standard as of late 2022.

Once you know this short history, it's easy to see how it follows the same gradual improvement approach of a nascent FinOps practice:

1. You start by checking to see the total you're spending by service before paying the cloud provider. (Invoices)
2. Then you can probe into which teams or products are driving that service, so you can do showback or chargeback. (CAR)
3. Then you begin to wonder when resources are being used. (DBR)
4. Unsatisfied with that, you realize you want to pinpoint specific resource rate application, as well as identify where your cost allocation gaps exist. (DBR-RT)
5. With a growing selection of cloud service provider products being consumed, you seek more granularity to filter for specific details in your cloud usage. In addition, you look to make more of your FinOps work programmatic and to answer questions about the ROI and value of your commitments such as RIs. (CUR)
6. And finally (or at least for now, since FinOps is always evolving), offerings like Amazon Billing Conductor enable you to adjust rates, slice bills, and deliver custom billing data for individual specific purposes.

That evolution also maps to the journeys of Cloudability, one of the earliest cloud cost platforms. Their platform in 2011 was just a set of scripts that logged into AWS (using root credentials since IAM—identity and access management—didn't yet exist), screen-scraped invoices, parsed out the total spend line items, and then sent a simple daily email with the numbers. To this day, the daily mail feature is still the beating heart of the platform, driving a feedback loop for teams. But as the billing data has become more complex, all FinOps Certified platforms (like CloudHealth, Spot, CloudCheckr, and others) have adapted and evolved with functionality to match.

We told the preceding story through the lens of AWS billing data simply because, at the time of writing, it was the most mature. But rest assured, each of the other providers is undergoing (or has undergone) a similar journey of billing maturity. The multicloud journey unfortunately comes with even more complexity. Azure for a long time has supported different APIs based on the type of contract you have with Microsoft (though some of these APIs are being deprecated and consolidated as of late 2022), and the same information is not always available through each. As of 2022, Google Cloud still does not have fully granular resource-level data for all services via the same mechanism, but there are various workarounds. But all cloud providers are continuously updating how they provide their cost and billing data, and they are making modifications to support more and more new services, new billing constructs, and new pricing models.

The Importance of Hourly Data

You might wonder why you need to consider hourly (or per-second level) data and resource-level granularity. Isn't daily or weekly granularity at the tag level enough? The simple answer is no. And definitely not once your FinOps practice matures.

Hourly data is required to do higher-level FinOps functions like commitment-based discount planning. SP/RI/CUD purchasing is based on how many of a certain type of resource you're running over a certain period of time in relation to the break-even point for not reserving the resource. If you count up resources over a month, you don't get that important detail. To determine how many resources you need, you have to look at how many were running each hour (or second), along with the waterline of usage.

We won't go into detail on that here (this story is still in the early maturity stage), but remember down the road that the fine-grained visibility that AWS, GCP, and Azure et al. provide is critical to access in a world of variable resources that can exist, and be charged, for only a matter of seconds or minutes.

A Month Is Not a Month

Imagine that a company has adopted a new cost optimization initiative at the beginning of the year. It starts off on January 1 with a list of cost-cutting actions: "We turned off unused instances, we rightsized others, and we bought some Savings Plans." The following month, the cloud bill drops 10%. Success!

But wait. If you divide the number of days in February by the number of days in January, you get 90%. That may seem painfully obvious, but you can't even begin to count the number of times that the delta between February and other months has falsely convinced a team that they've been effective in reducing costs. Then, inevitably, on March 31, there's a panic because costs are up 10% month over month.

It's worth repeating that *cloud billing is all about time*. Cloud billing doesn't use a fixed month like many finance teams are used to. It's much more granular, and if you want to make an apples-to-apples comparison, you need to look at the same lengths of time, whether that's 10 days or 10 hours.

This is a big jump when you're coming out of a world of waterfall amortization schedules where you divide the year's spending evenly by 12. And old habits are hard to break. We've actually seen cloud-challenged finance departments criticize properly calculated hour-level amortizations as being off by 10% because they divided their amortization by 12, when they should have taken the amortized cost per second (or hour) and multiplied it by the number of hours used. It's yet another example of how FinOps requires a new mindset.

A Dollar Is Not a Dollar

While we're on the topic of apples to apples, remember that the rate for a specific resource type in a specific row may be different than the same resource type in a different resource time period. Unlike with hardware you own or lease, where you can set a consistent rate for resources across a time period, cloud rates can vary wildly based on whether a discount was applied by the cloud provider.

Further, amortization of early prepayments of SPs or RIs may change the equation even more, and you need to decide whether to factor these into the rates your users are seeing. We recommend including them, as that means the amount shown will better represent the amount you later chargeback. This, of course, assumes you've gotten to that stage in your allocation journey.

If you chose not to include the amortized prepayments, your teams might think they're spending less than they really are, especially in the case of the up-front RIs that are offered by AWS and Azure. When those aren't included, the row in the billing data for the applied portion of usage ends up at $0.00—even though a resource was used that cost you money that was paid in advance. You'd make your teams feel great about their cost reduction, but you'd also give them a false sense of efficiency.

All cloud service providers also offer what is called a *Free Tier* for some services, sometimes as a promotion, and sometimes on a continuing monthly basis. It's generally a small amount for large cloud users, but this is another way this variance in rate can cause confusion. There is often no consistency to which workload gets the benefit of the free usage every month, and this can lead to your teams seeing small, confusing variation in their month-on-month costs.

Remember that rates and costs can change without a team making any changes to the infrastructure.

Two Levers to Affect Your Bill

Remember, the simple formula for cloud spend gives us two basic levers that affect cloud spending: *Usage* and *Rate*.

$$\text{Spend} = \text{Usage} \times \text{Rate}$$

This is going to be a key part of deciding both how to optimize and who in the organization takes optimization action. Let's look first at the *how*, and then we'll cover the *who*.

The first lever is to reduce what you use. This is called *cost avoidance*, and you might do this by terminating idle resources, rightsizing oversized ones, scaling down the number of resources running during off-peak times, or shutting things down completely over nights and weekends.

The second lever is to pay less for what you use. This is called *rate reduction*, and you do this by taking advantage of cloud billing constructs such as Savings Plans (AWS or Azure), Reserved Instances (AWS or Azure) and Committed Use Discounts (GCP). There's also volume discounting based on usage (e.g., the Sustained Use Discount from GCP) or custom pricing programs that some cloud providers offer to large-scale spenders. Finally, some companies also use spot instances or preemptible instances, which can be useful if they're willing to engineer around a potential sudden loss of the resource. Whichever way you choose, they all lead to paying less for what you use.

Who Should Avoid Costs and Who Should Reduce Rates?

There's been debate around who's responsible for each process when it comes to optimization tasks. After over a decade each of working with hundreds of companies who have done FinOps poorly and a (much smaller) number who have gotten to more mature stages of FinOps, we have a firm opinion on this:

> The most successful FinOps practices decentralize using less (i.e., avoiding costs) and centralize paying less (i.e., reducing rates).

The decentralized decision makers responsible for driving the majority of cloud spend are the application owners themselves who are directly making infrastructure choices. They're best equipped to make a decision about shutting things down, resizing them, or changing the shape of the demand. Due to their intimate familiarity with the workload needs, they can look at utilization data and determine whether the instance that appears to be idle needs to stay around or can be terminated. The ability to make infrastructure decisions cannot be centralized effectively in a large, distributed organization. Give the engineers/application owners the data and the power to make the right decisions for their business objectives.

On the other side of the coin, these same application owners are not typically in the best position to buy RIs or CUDs. Worse, they sometimes misunderstand the financial mechanics of how those work, or they don't understand important related concepts like the company's cost of capital. A centralized FinOps team can look at the entire cloud estate for opportunities to save. Even better, they likely have a procurement or financial analysis skill set that understands the nuances of cash flow analysis and are versed in both the technology needs and the financial constraints of the organization. Chapters 16, 17, and 18 provide a deeper analysis of the various commitment-based discounts available and strategies to use them effectively.

Measure the central FinOps team on metrics tied to increasing reservation/commitment coverage, reducing unused vacancy, and taking advantage of volume discounts or negotiated pricing.

Centralizing Rate Reduction

One of the FinOps principles is that a centralized team enables FinOps, and centralized rate reductions are one of their key responsibility areas. Rate reduction offerings can be complex to understand, and cloud service providers are constantly innovating on their offerings. While we were writing the second edition of this book, both Google Cloud and Microsoft Azure announced completely new commitment-based discount programs in the form of Flexible CUDs and Saving Plans, respectively. Trying to get each distributed team to spend time learning how each new discount program works and how best to track, optimize, and operate them isn't effective.

All of the units of cloud resources that your teams expect to run at a reduced rate won't come from one team. Enterprises at scale run multiple products and projects that require hundreds to thousands of cloud resources, and compiling all of the reservations into one centralized means of monitoring increases shared coverage.

Since a single SP or CUD can apply to multiple resources—and in the case of AWS, across multiple accounts—having individual teams manage their own rate reduction programs often results in reducing overall coverage rates, covering too much in one area, and ultimately unused capacity of purchased commitments. The highest level of savings is achieved when they are managed by a central team with an overall view of what's needed. As we'll cover in Chapter 18 on commitment-based discount strategies, there's more to consider than just resource type when making commitments.

For instance, one team may have high resource usage during the day while another has high resource usage during the night. Based on their usage, it probably doesn't make sense for either team to make commitments individually. But overall, there's a consistent base of resources running across the 24-hour period. The central team identifies the opportunity to commit to a reservation and save both teams on the rate they pay for resources. An image you can use to visualize this behavior is two sine waves that have been phase shifted—see Figure 5-2; as one decreases the other increases and over time the area is largely filled with some activity.

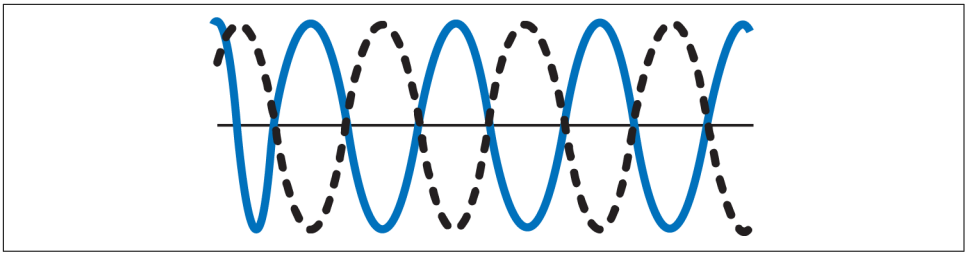


Figure 5-2. Phase shift overlap

At the more advanced stages of FinOps maturity, keeping discount coverage rates up and unused capacity down requires constant management. In addition, the initial purchases can be complicated to get right (we've seen millions of dollars wasted in a single bad commitment). The FinOps team lowers the potential for waste while optimizing coverage for all.

Why You Should Decentralize Usage Reduction

Conversely, another FinOps principle states, "Everyone takes ownership of their cloud use." With rate reduction now strategically centralized, cost avoidance (usage reduction) becomes an activity-promoted approach across all teams in your organization. At the enterprise scale, cloud costs can comprise hundreds to thousands of operations per day by various teams and various engineers supporting the services that your enterprise runs. They're the beating heart of your innovation machine.

To keep them moving quickly, start by being proactive and establishing a lightweight education program for your engineering teams. Advocate for cost efficiency during architecting and explain methods engineers can use to deploy cloud in a lean way, then grow their resources as their workloads require. Some examples include effective use of scaling architectures and of spot instances, loosely coupled architectures, serverless or containers, implementing policies around data retention before teams begin storing data, embedding cost estimation of changes before being released, and flagging optimizations during release reviews (pull requests and/or change review meetings), such as updating the use of old generation instances defined in infrastructure as code.

Though decentralized teams are often best equipped to make decisions around infrastructure changes, it's common for a central FinOps team to provide optimization data and recommendations for them to consider. They do this by gathering usage and utilization data and providing visibility into alternatives for resources that better match workload needs. Ideally, those recommendations are pushed out to the teams on a regular cadence, on a threshold alert basis (for an early stage practice) or into developer sprints via ticketing systems like Jira or ServiceNow (in an advanced stage

practice). We'll cover the key criteria for those recommendations in deeper detail in Chapter 15.

This model ensures that your resource owners—who have a better understanding of how a resource is used, what depends on it, and whether there are any business reasons for why changing it would negatively impact your services—have a chance to review all recommendations before they're implemented.

The FinOps practitioner empowers engineering teams by giving them the insight and data access to make the best technology decisions in near real time. They introduce both savings potential and spend impact as cost metrics alongside the other metrics the engineering teams already track.

Conclusion

Cloud billing is complex. Providing access to the right data granularity and scope involves work and trade-offs. But that very complexity gives you a lot of power to both analyze what's driving spend and empower your teams to own their optimization decisions. Even when you're using a FinOps platform to automate reporting and insights, you need to get to know the format of your cloud provider's data so you can draw the proper conclusions from what you see. Aim to standardize your organization on a consistent cost metric format and mechanism for reporting on it. That will likely be an amortized rate that factors in any custom discounts you may have with your cloud provider, amortized prepayments and any support costs it charges. Without this standardization, you'll have lots of confusion because your teams will be looking at the numbers differently.

To summarize:

- Billing data is time-based in most cases, and so it requires constant attention to make sure running resources continue to provide value.
- Detailed analytics of your billing data are possible only with a deep understanding of that billing data. Learn the raw data to become an effective FinOps practitioner.
- Small changes in your cloud bill can add up quickly, so track changes using automated anomaly detection and variance reporting to identify what is trending up.
- The simple formula for your bill is *Spend* = *Usage* × *Rate*. This gives you two levers to optimize your bill: using less and paying less for what you use.
- Decentralize reduction of usage (using less), while centralizing reduction of rates (paying less for what you use).

- A central team is better equipped to manage commitment-based discounts due to their ability to look across the entire cloud estate and the complexity of managing a large portfolio of commitments.
- The decentralized teams are empowered with usage optimization data and recommendations by the central team, as these decentralized application owners are the ones best positioned to make infrastructure change decisions.

We've laid the groundwork by introducing FinOps, why you need it, how it requires a cultural shift, and, finally, what insights the cloud bill provides. The next chapter covers the FinOps Framework. We will describe what the framework is, how you use it, and the structure of each component.

CHAPTER 6
Adopting FinOps

FinOps is an infinite game. By this we mean that you're never really done at getting better at it. And you have to start somewhere.

> *In finite games, like football or chess, the players are known, the rules are fixed, and the endpoint is clear. The winners and losers are easily identified. In infinite games, like business or politics or life itself, the players come and go, the rules are changeable, and there is no defined endpoint. There are no winners or losers in an infinite game; there is only ahead and behind.*
>
> —Simon Sinek, *The Infinite Game* (Penguin UK, 2020)

One of the most common questions we hear from the community of practitioners is "How do I get my executive to buy in to the practice?" and, then, "How do I get the organization to adopt the practice across the broad swath of personas critical to its success?"

This chapter works through common approaches to gaining proper organizational support from both the technical and financial leadership of the company. Without that, nascent FinOps practices often get stuck in a reactive midstage maturity battling other priorities in the organization. The chapter also looks at the processes of the *FinOps Driver*, the person who recognizes the need for the practice and champions it internally.

Figure 6-1 outlines key personas to inform and milestones to accomplish as you help build the practice and culture at your organization.

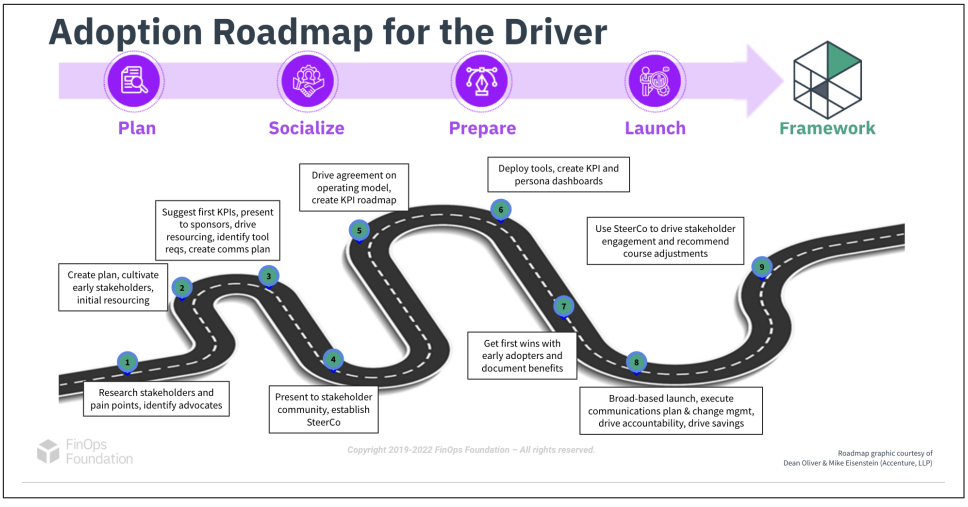


Figure 6-1. The FinOps Adoption Roadmap, original source Mike Eisenstein and Dean Oliver (Accenture, LLP)

A Confession

While writing the second edition of this book, my coauthor Mike Fuller confessed to me: "In hindsight, we [Atlassian] weren't serious about FinOps when we wrote the first edition of the book." Yes, FinOps was one of his areas of responsibility, and he had spent years working and thinking on the patterns of FinOps while managing Atlassian's cloud spend during their massive migration and growth in cloud, but it wasn't his full-time job.

In fact, it was no one's full-time job at the company. There were no top-down C-level codified company goals around FinOps outcomes. Target setting was inconsistent and they were not leaning aggressively into optimizations and processes so much as focusing on low-hanging fruit. He knew what they needed to do and how to get them there, but it was simply not the company's highest priority. And that was OK. The company was doing *informed ignoring*—making an intentional decision to focus on other priorities like completing migrations and delivering products more quickly.

At first, my jaw dropped. I had been taking notes on Mike's advanced thinking around cloud cost management since we first met in 2014 and he—in his quietly authoritative fashion—reset my thinking around a number of core areas like RI strategy and data sources.

However, the more I thought about it, Mike's "confession" about his own journey (and by extension Atlassian's) into FinOps maturity was just like everyone else's. Invariably, you start FinOps as a light practice—or an early maturity one—and over time, as cloud spend grows and company goals align, you transition to taking it more

seriously. Mike was always personally obsessed with FinOps, but he had a day job (at the time he was working within Atlassian's CCoE), and his company (like most) took some time to get to the point where they decided to heavily invest in staffing the FinOps functions and roles needed to promulgate the practice. He is the classic FinOps driver archetype: it starts with a noisy individual, then over time it turns into a company focus with broadly prioritized, executive-sponsored goals.

As we dug into this thinking, we realized that there are generally a couple of levels of practice maturity. This is important to unpack at the start of this chapter, as the pitch you will make to leadership will be very different depending on which level the company is on. Before you pitch FinOps internally, you need to take an honest look at where the company sits in its FinOps journey, which is typically tied to its cloud adoption journey and which outcomes it is most actively prioritizing.

FinOps isn't about saving money, it's about making money through data-informed decisions about your cloud technology investments. If your top priority is to get out of your data centers as soon as possible, or to leverage cloud to transform your technology and drive more revenue, then you're likely not going to get a lot of buy-in to focus on long-tail optimizations; you'll mainly get support to accelerate migrations.

Though Atlassian was practicing informed ignoring, it didn't mean they weren't doing anything. It meant they were selectively leaning into capabilities they saw as critical and being passive about others. Cost allocation was seen as critical to accountability and was aggressively pursued, whereas commitment-based discounts and rightsizing were being done more passively.

There hit a point, though, where Mike knew he needed to convert the passive practice to an actively managed one.

Different Executive Pitches for Different Levels

To ensure you get the most success, let's turn again to the FinOps maturity model discussed in Chapter 1. At the early stages, you'll have a certain pitch, based on the outcomes you are looking to achieve. To do that, you must assess where your organization really is.

In the less mature stages, you're likely doing things that are *FinOps adjacent*, such as reviewing spending or occasionally making optimizations, but you aren't doing them in a lifecycle, or running a framework (such as the FinOps Foundation Framework covered in Chapter 7) with capabilities, or adopting a playbook.

Mike later added:

> We weren't doing FinOps in a structured way like we are today. We had been doing it for years in a more ad hoc one. In the middle stages of our FinOps maturity we started setting goals on all the framework capability areas and set targets and thresholds for those capabilities, tracking which ones we aimed to improve, measured the results of our efforts, and reported it out.

Starting Pitch

> *What the wise man does in the beginning, the fool does in the end.*
>
> —Warren Buffett

Here are some questions you'll want to have answers to when pitching initial FinOps adoption:

Why should you start doing FinOps?
: Consider the first few chapters of this book for help in sharing the *why* to get started.

What are the benefits to each executive persona involved?
: Each executive is focused (and bonused) in different ways. If individual executives know the specifics, this will help them build their own cases for supporting the initiative. Later in this chapter, we'll break down the individual executive challenges, motivations, and benefits between CEOs, CFOs, CTOs, etc.

What are the benefits to the business in terms of your cloud adoption and the outcomes the organization will receive?
: Consider the main outcomes of the FinOps Framework (which are covered in Chapter 7). Also look closely at your cloud adoption to determine the balance between such benefits as cost allocation, optimization, etc. Cloud adoption goals prescribe where on the *Iron Triangle* to focus efforts, as discussed in Chapter 26.

How can FinOps help unblock the pathway to other important initiatives like security, GreenOps, and sustainability?
: We're starting to see strong overlaps between GreenOps and FinOps teams. They share a set of common underlying goals. Tying sustainability goals to FinOps can help with funding for the ask; two birds, one stone. See Chapter 19 to understand more about how sustainability and FinOps intersect.

Why do existing financial frameworks not solve the need?
: Existing IT financial frameworks often work in parallel to a FinOps practice by solving a different set of needs in the business. See Chapter 25 to understand how other common frameworks may work together (or not) with FinOps. It's important to have answers to this question when you encounter those more

familiar with traditional IT frameworks who are not used to dealing with massive variability and granularity of spending data unique to the cloud at scale.

Since this is a starting pitch, you won't yet have a FinOps framework structured that clearly outlines the different capabilities you need within your organization. Instead, you are presenting high-level outcomes that answer questions such as: can you understand your costs and usage, are you able to identify optimization opportunities, etc.?

Advancing Pitch

Here are some things you'll want to include when pitching to grow a practice's maturity.

A good starting place is to measure how successful your organization is in individual FinOps capabilities. This allows you to define and pitch where you need to go next. This stage of pitch is focused on how you can invest more heavily into specific areas with specific outcomes you expect to achieve.

Each of the capabilities in Table 6-1 should have a measure of success and a set of outcomes, which is covered in greater detail in Chapter 7. Focus on telling the story of how to hit the outcome you need to hit the target measure of success. Then connect these outcomes to the asks required from the business to achieve them, such as:

- Time commitments from other teams
- Additional money to be spent or payments to be made up front
- Added headcounts in either the FinOps team itself or other teams across your organization

Table 6-1. Simplified assessment of the maturity of framework capabilities in an existing practice

Domain	Capability	Maturity		
		Early	Middle	Advanced
Understanding cloud usage and cost	Cost allocation		Current	Grow
	Managing shared costs	Current	Grow	
Performance tracking and benchmarking	Forecasting		Build	
Cloud rate optimization	Managing commitment-based discounts		Current	Grow
Cloud usage optimization	Resource utilization and efficiency			Build
Real-time decision making	Measuring unit costs	Build		
Organizational alignment	FinOps intersection with other frameworks	Current		Grow

Here are some examples of capability outcomes you'll want to highlight with additional investment in the practice:

Cost allocation
: Improve the percentage of costs being correctly allocated to teams.

Forecasting
: Increase reliability and predictability, etc.

Measuring unit costs
: Identify teams that can start measuring a business value metric for a set of products and tie it back to their reporting to use in their value conversations.

FinOps intersection with other frameworks
: Provide alignment between teams managing other finance practices and FinOps within the organization, reduce duplication of work, and improve consistency of reporting on cloud spend.

Budget management
: Improve the ability to both stay inside of forecasted budgets and to "shift left" proactive budget management such that engineering teams proactively predict the cost impact of changes and raise awareness to budget leaders when material impacts are expected.

State of FinOps 2022 data showed early stage teams tend to be 1–3 people, whereas advanced stage companies tend to have upward of 13 dedicated people helping to enable FinOps in the organization. The most recent survey data can be found on the FinOps Foundation site.

Going back to Mike's case, improving cost takeouts was the original pitch. But in the advancing stage, it has become all about engineering efficiency. He looked to answer the question of "How can we keep engineers from getting bogged down and increase the velocity of releases?" He knew that improved engineering cost efficiency would allow his technology leader to save money and enable funding of additional projects and headcount: "Give us some money now to improve our FinOps practice, and you get more money later to do more innovative things with more engineers."

Overspend can limit technology team velocity. Too often when teams overspend, they are told to pull back on feature development and focus on reducing costs by burning down a backlog of waste. Ideally, teams are structured to keep ahead of the problem from day one and are cost architecting as they go to prevent slowdowns. It's a classic maintenance and KTLO (keep the lights on) approach versus getting bogged down in firefighting. This enables predictable, smooth engineering workloads instead of large spikes of effort that take their attention away from releases when the waste gets too high. This concept is often a key part of your pitch to advance your practice.

FinOps can present a large palette of things to solve in the organization. Each will be different. The complete picture could go as far as influencing what the business decided to invest in. Table 6-2 shows some examples of the focus areas to present to different personas.

Table 6-2. Examples of focus areas that FinOps can help solve for different personas

Persona	Focus
Business leaders	Leverage cloud to be more competitive
Engineering	Innovate while increasing cost efficiency and speed of delivery
Finance	Understand, allocate, and predict cost accurately

All are interested in cost savings to some degree, but each has different core motivations. Today, fewer companies are starting with zero FinOps; most have some level of FinOps activity already happening. So you need to paint a picture of where you are today, where you are going, and the benefits of that long journey.

Sample Headcount Plan for Advancing a FinOps Team

Table 6-3 is a sample headcount plan as part of an advancing FinOps pitch made to a technology executive. It includes additional headcount for key areas such as data engineers to ingest/normalize the billing data and related feeds, analysts to interrogate the data and prepare reporting for the organization, cost engineers who embed within service teams to uplevel their usage optimization skills, automation engineers to connect various FinOps services and provide templates to the organization, and leads who manage the FinOps team itself.

Table 6-3. Sample hiring plan for a CTO-reporting practice

Financial year	Data engineers	Analysts	Automation engineers	Team leads	Funding required
FY22	2	1	1	1	
FY23	2	2	2	1	+2
FY24	3	3	2	2	+3
FY25	4	5	3	2	+4

Your advancing FinOps pitch should include the outcomes you expect to see happen once you get the resource needed. For example, the organization needs $X million of commitments to reduce rates, three new headcounts, and commitments from these three other teams (finance, engineering, and procurement) to build out a business value model.

Stories from the Cloud—RJ Hazra

RJ Hazra, SVP and CFO Technology at Equifax, shares how their own adoption journey changed the focus of the application teams to focus on cost: "They were primarily busy moving workloads, not looking closely enough on the best outcome on the business. We had a lot of success by taking engineers who had worked on cost-optimized central platforms and embedding them to business unit teams, which shifted our focus to say that FinOps is now everyone's responsibility. We publish the playbooks, we build and architecture diagrams, you follow the patterns we set, and we evaluate you against those. If they don't follow them, we slow you down."

Pitching the Executive Sponsor

As of 2022, State of FinOps data shows that the majority of FinOps teams report to a technology executive like the CIO or CTO. A growing number report to a CFO for Technology or a related hybrid role. The first part of your pitch is to paint the picture to the CTO (or the right CxO for your org) where you stand today. This includes covering what reporting visibility is in place as well as an assessment of the company's maturity along key capabilities of the FinOps Framework. We recommend starting with the open source FinOps assessment available on the FinOps Foundation website.

Most importantly, you need to portray what you expect to change in the company's culture and ways of working once the next phase of the practice is implemented.

Here are some questions you can answer in your pitch:

- How does the daily work of an engineer change?
- What programs and rituals does the organization need to introduce?
- How do engineering managers start to think about FinOps?
- How does the organization ensure they have the right resources?
- Digging into visibility advantages, what unit economics and reports could the organization drive?

Here are some more questions that begin to model some specific scenarios:

- In what mix should the organization plan to achieve cost avoidance: by focusing more on usage reduction or more on rate reduction?
- How many more SPs or CUDs could the organization have if they had more time and people to focus on maintaining them properly?
- How much more you could have saved in the last year if you were able to manage commitments and resource utilization more effectively?

- What do you need to change so when you perform a lookback a year from now, your efficiency is markedly improved?
- How much more usage reduction could the organization do if they had whole company goals or better executive support for such changes?

The pitch should cover what it really means for the company. It is not simply about the amount of money it costs, the number of people needed, and the dollars saved. At Atlassian, as it is in most companies, it was important for the CTO to think through the impact of adopting FinOps on the engineering team (a technology company's most important asset) before moving into the value proposition of savings and efficiency it would drive.

So before making the pitch to the Atlassian CTO, Mike got feedback from relevant teams. He interviewed finance about pain points they had around allocation of spending and accurate forecasting of variable spending. He talked to engineering service owners about how much visibility they currently had into cost and what they would need in order to be able to explain to leadership what had happened when anomalies occurred. Through all the conversations, he looked at how the central FinOps team he was pitching could help.

These conversations also began to prepare the teams for the changes that were coming: service owners would be expected to know what their costs were and to be prepared for discussions when spending changed. He also began to socialize the critical strategy of putting data in the path of the engineer (see Chapter 8 for details on this concept). He began to set expectations for what each level of engineering would do once the more structured FinOps goals and practice were rolled out.

In Mike's case, funding came from the CTO, so the conversation was centered around engineering gains. When pitching to a CFO, the focus should be more on ensuring better reporting of costs via cleaner allocation, more accurate forecasting, and lower variances against budget targets that could be improved with more investment.

Playing to Your Audience

When proposing the adoption of a FinOps function within an organization, brief a variety of personas among the executive team (engineering leadership, finance leadership, etc.) to gain approval, buy-in, and involvement in conducting FinOps and achieving its goals.

Each executive team persona is described below, in terms of their goals, concerns, key messaging, and useful KPIs. By understanding the motivations of each executive persona, a FinOps champion will be able to describe the value of FinOps more effectively, minimizing the time and effort to gain alignment.

Key Personas That the Driver Must Influence

Figure 6-2 shows how the FinOps team and driver persona sits initially in between all the various stakeholders helping to translate needs and build bridges that allow them to work more closely together autonomously.

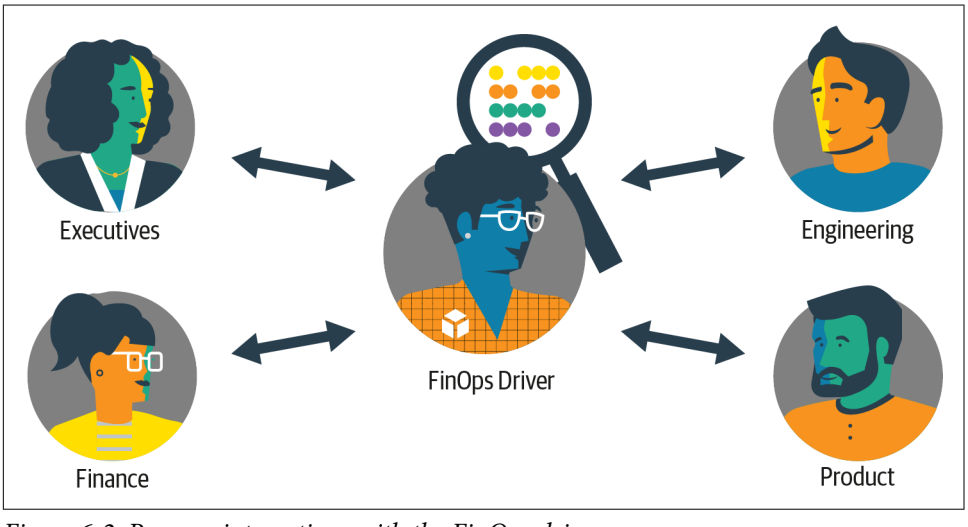


Figure 6-2. Persona interactions with the FinOps driver

For a more comprehensive look at various personas and roles involved in building a FinOps practice, see the individual personas motivations in Chapter 3. Parts of the following sections are adapted from the Adopting FinOps open source project.

CEO Persona

When it comes to a FinOps practice, the CEO's goal is to ensure that cloud investments are aligned with business objectives. See Table 6-4 for examples.

Table 6-4. Common CEO business and FinOps objectives

Objectives	Frustrations	Key metrics	FinOps benefits
Accelerate company growth YoY Strategic competitive advantage Faster time-to-market Deliver innovative, market-leading solutions cost effectively	Unpredictable, sometimes chaotic, cloud spend Unable to see link between engineering initiatives and business objectives Unsure of the return on their company's cloud investment	Revenue growth Gross margins COGS (cost of goods sold) Unit cost economics	Manage risk Connect engineering decisions with business outcomes Predict how cloud spend will grow as the business grows Guide organization to make good cloud investments

CTO/CIO Persona

Leverage technology to give the business a market and competitive advantage. See Table 6-5 for examples.

Table 6-5. Common CTO/CIO and FinOps objectives

Objectives	Frustrations	Key metrics	FinOps benefits
Accelerate company growth YoY Strategic competitive advantage Faster time-to-market Deliver innovative, market-leading solutions cost effectively	Extreme pressure to either justify or bring down the cloud bill No guardrails on spend Unsatisfied engineers Business continuity Reliability	Revenue growth Gross margins COGS Unit cost economics Time to market for new features/product Engineering productivity Track R&D versus production spend Maintain operations within budget	Predict how cloud spend will grow as the business grows Drive organization to make good cloud investments Enable engineering organizations to gain more freedom to utilize newer cloud technologies and deliver solutions to market faster

CFO Persona

Increase accuracy and predictability of cost reporting and forecasting. See Table 6-6 for examples.

Table 6-6. Common CFO and FinOps objectives

Objectives	Frustrations	Key metrics	FinOps benefits
Cost visibility and granularity, and accuracy Overall cost-per-unit/COGS reduction Managing costs during growth, disproportionate cost reduction during flat/declining periods	Unpredictable, sometimes chaotic, cloud spend Unsure of the return on their company's cloud investment Cost fluctuations and innovation do not align to budgeting cycles	Revenue growth Gross margins COGS Unit cost economics Predictability	Increased accountability for cloud cost Increased reliance on budget and forecast models Direct impact on company bottom line

Engineering Lead Persona

Increase speed of delivery while also being cost-effective and enabling innovation. See Table 6-7 for examples.

Table 6-7. Common engineering lead objectives

Objectives	Frustrations	Key metrics	FinOps benefits
Drive accountability to engineering teams that are responsible for the application/services Provide guidance to engineering teams to have a cost-effective application/service by identifying anomalies and best practices Work together with engineering teams to identify rate reductions and possible cost avoidance Cost allocation	Unsatisfied engineers as their workload keeps rising Long delivery cycles Cannot predict the impact on the budget Difficult to identify service or application ownership Cannot predict the costs closely enough for developing new features and products	Revenue by infrastructure costs Cost per deployed service and service utilization rates Showback and chargeback of IT costs to the business	Increased visibility to cloud cost Connection to cloud cost and unit economics More accountability for utilization Incentive toward solid architecture principles to factor efficiency

Roadmap for Getting Adoption of FinOps

We've provided this starter guide to help you build a presentation to inform other teams, teammates, and stakeholders about the benefits of building a FinOps practice. It includes preparation steps, accountability and expectations per role and persona, a detailed roadmap, and more.

The stages below are based on the open source sprint outputs of the Adopting FinOps working group at the FinOps Foundation, which included core contributions from Mike Eisenstein (Accenture), Idaliz Baez (RealOps.io), Natalie Daley (HSBC), Rejane Leite (Flexera), Mike Bradbury (JunoCX), Tracy Roesler (dbt Labs), Bailey Caldwell (McKinsey), Rich Gibbons (ITAM Review), Erik Peterson (CloudZero), Nick Grab (Ellix GmbH), Anthony Johnson (Box), Kim Wier (Target), Melvin Brown II (US government), and Mandy van Os (TechNative).

Stage 1: Planning for FinOps in an Organization

As stated at the beginning of the chapter, FinOps is an infinite game. You're never really done. It's crucial that you get started as soon as possible and start taking baby steps toward maturing the capabilities most important to your business. The goal of this stage is to have a suitable plan for FinOps transformation in your organization that has been adapted to the resourcing you have available and the goals that you've outlined as being most important.

Do your research

Seek out the right stakeholders within the organization. As an individual looking to bring FinOps to your organization, you will need senior-level sponsorship as well as cultivated supporters to build momentum. Here are some steps you can take:

- Research your possible advocate/champion/executive sponsor, and have one-on-one conversations with them using a customized FinOps introduction deck or targeted interview questions to determine adoption strategy.
- Research pain points being experienced by the organization during your conversations, such as cloud costs breaking business cases, general perception of cost overruns, lack of cost visibility by cloud consumers, etc.
- Research impacted groups, teams, and individuals during your conversations. Who is affected by the pain points?

Create a plan

Paint a picture of the future state. This should appeal to the business (business acceleration due to valuation of cloud technology, unit economics improvements, etc.). Here are some steps you can take:

- Identify tool requirements. Determine if existing owned tools can fill the needs of the plan.
- Identify an organizational home for the FinOps function. This may be in a CCoE, in finance, or in IT. Depending on the complexity of the organization structure, creating a dedicated FinOps team might take a phased approach. Some organizations might (1) set up a cross-functional transformation program office and create workstreams/working groups, (2) create a FinOps function as part of the extended Cloud Business Office/CCoE, and (3) evolve into a dedicated cloud FinOps team.
- Identify candidate early-adopter teams who can be evangelists.
- Identify KPIs to measure the FinOps function and identify ways to measure engagement and performance of stakeholders like business units and application teams. These KPIs are preliminary and will evolve during Stage 2, but it's important to have a starting set.
- Prepare a communication plan that will be used in Stage 2.

Stories from the Cloud—Jason Rhoades

Jason Rhoades at Intuit shares the importance of delivering big, quick wins early on to increase investment from executives:

> Everywhere is different of course, but for me at Intuit, actual wins have preceded all incremental FinOps investments. FinOps as a theory may be hard for some execs to grasp. Or you're stuck with CTO seeing it as a pure tech effort and CFO as pure finance. Painting the picture of "we saved $XM last month with our new FinOps prepay effort, and still have $YM more to go after" or "we traced the root cause of

> cloud cost spike to an action made by a particular team, and could have prevented it with investment"—when you're in a greenfield situation, those demonstrated and relatively-easily-achieved early results paint a more tangible/visceral and executive graspable picture of how FinOps can and will add value in the long run. Leadership is much more likely to back an effort that has already shown demonstrated, quantified success, than one that is just talking about potential. Prove the value and you'll get the backing you seek. The communication plan shouldn't over-index on the details of FinOps but rather tell the story of what FinOps can do for the business.

Gather support

Bring cultivated supporters together and explain why it is important to adopt FinOps:

- Highlight current state, pain points, and other issues.
- Identify threats and provide scenarios that could happen if action is not taken.
- Demonstrate what maturing FinOps would look like for the organization.
- Examine opportunities that should be, or could be, exploited.
- Present a roadmap for implementing the plan.
  - Get feedback from the executive sponsor and adjust as needed.
  - Include initial team size, budget, and timeline for initialization.
  - Include the value proposition (e.g., ROI such as the cost of having a FinOps function versus an ongoing reasonable cloud overspend).
- Present to other stakeholders, supporters, and a pilot set of people unfamiliar with the project such as business unit leads.

Perform initial resourcing

Successful FinOps requires carving out dedicated resourcing with relevant skill sets:

- Request support from the sponsor to recruit other executive leaders as sponsors.
- Form a change coalition made up of true organizational influencers and stakeholders.
- Get budget approval for headcount.
- Choose native, third-party, or homegrown tooling depending on need.

Stage 2: Socializing FinOps for Adoption in an Organization

FinOps is a cultural change that requires buy-in from many different teams and personas across the organization. As such, communication and feedback loops designed to encourage adoption of the practice are key. The goal of this stage is to socialize the

FinOps plan created in Stage 1 to all the relevant stakeholders. The FinOps pitch deck below helps communicate this in a clear, easy to read, and quick manner.

Communicate value

How key stakeholders promote FinOps in the organization:

- Communicate the values that are central to the change.
- Share a short summary of what you *see* the future organization looking like.
- Share high-level roadmap.
- The *What Is FinOps* pitch deck available on the FinOps Foundation website is a good starting point for this pitch, but it must be customized for your organization to touch on your plan.

Gather team feedback

Create FinOps conversations with identified impacted teams, such as finance lead(s), product lead(s), and lead engineer(s), to:

- Provide an understanding of what FinOps is.
- Understand their issues and explain/educate on how FinOps could help them.
- Discuss proposed KPIs and adjust per conversation feedback.
- Establish interaction model between FinOps and key partners (IT domains; controllers; app teams).
- Identify future members during socialization for CCoE and executive steering committee.

Define initial FinOps model

Your model should include resources, an operating model, patterns for cross-functional interaction, and related KPIs to report on progress:

- Customize the FinOps model (*inform, optimize, and operate*) to the organization.
- Identify the FinOps team with internal transfers where there is overlap with existing roles or individuals; fill remaining gaps via recruiting/contracting.
- Map the change network for FinOps across the organization—sponsors, influencers, adopters. Create a clear training and communications strategy that ensures full coverage of all impacted resources.
- Create a hub-and-spoke model to scale FinOps within very large organizations, detailed later in this chapter.

- KPI roadmap: Finalize the first set of KPIs and reports, and identify and plan for next-gen KPIs and reports.

Stage 3: Preparing the Organization for FinOps

Once you've socialized and gathered support for the cultural changes, now comes the work of assessing and engaging the various parts of the business. The goal of this stage is to begin starting the FinOps flywheel in your organization so you can begin maturing across the various capabilities and domains of the FinOps framework.

Assess FinOps readiness

Performing an assessment of the organization's maturity in key capabilities often includes the following:

- Define tags, metadata, and organizational taxonomy.
- Deploy, configure, and smoke-test tool(s).
- Finalize the first wave of KPIs. KPIs/business adoption metrics can evolve over "adoption periods" to create a mentality of gradual FinOps maturity and not push full maturity in one go. This will allow for less mature teams and executives to not be "scared off" and do it step-by-step.
- Define usage and spend thresholds for alerts and report limits.
- Define and prepare persona-based self-service dashboards. These should show important metrics like the first wave of KPIs, cost allocation, budget anomalies, optimization recommendations, and other views of interest to stakeholders.
- Prepare forecasting model with unit cost calculations included. At this point, this is likely just a spreadsheet.

Engage stakeholders

You're now ready to begin ongoing cadences and processes to implement your model:

- Determine business unit appetite for commitment levels (total cost for enterprise discount negotiations, RI/SP/CUD/etc.).
- Engage early adopter team to get optimization wins (e.g., shutting down test environments or instances that are no longer in use to show material savings). These are important for socializing, rolling out, and winning additional adoption later.
- Get some additional early governance wins for getting FinOps implemented (e.g., tagging policy, lease-to-live automation, etc.).

- Start cadence of regular meetings. The FinOps/CCoE team should be talking on a regular basis with the business units, app teams, practitioners, and stakeholders to implement best practices and track KPIs.

Remember that if the organization has multiple business units operating from a federated cloud operating model, they will have differing levels of maturity. It is important that the change management considers this and allows them to adopt at differing paces.

Congratulations, you've begun to practice FinOps!

Type of Alignment to the Organization

We typically see three types of FinOps practice teams: centralized, decentralized, and hub and spoke.

With a *centralized* FinOps team, they are aligned directly to CIO, CFO, or CTO and act as a service to the businesses. They might be part of a CCoE, or they might not, if the organization doesn't have one. Business units with cloud spend interact with this dedicated FinOps team to collaborate on FinOps activities, and the dedicated FinOps team centralizes capabilities—such as managing commitment-based discounts—which most benefit from centralization.

Pros
: Authority, expertise, economies of scale

Cons
: Many stakeholders, enablement and education key, persuasion requires executive buy-in

With *decentralized* FinOps teams, they align to a specific business unit and serve that business unit's best interest. Each of the business units (BUs) with significant cloud spend would have a dedicated FinOps person within their group, but without an overarching central team driving best practices and repeatable patterns across the organization.

Pros
: Very agile, embedded with engineering teams

Cons
: Not maximizing rates, duplication of effort, susceptible to loss of key talent

We typically see decentralized teams when there is organizational recognition that FinOps needs to be happening but there is not a central executive buy-in to set up a formal centralized practice. For most organizations, it will be a transitory state at the beginning of the practice.

Hub-and-spoke teams allow for a combination of both centralized and decentralized.

Pros
: Key BUs have dedicated embedded FinOps resources in the spokes, central hub FinOps focuses on maximizing rates and enabling more spokes

Cons
: Teams require more investment, training, and hiring of the right people at the spoke level

Fidelity Investments has a long-standing and well-defined hub-and-spoke model for FinOps that they discussed at the 2022 FinOps X conference. A central team manages commitments and rolls up capabilities like forecasts from the various teams, but each major business unit also has dedicated and embedded FinOps experts funded by their respective business units.

Full Time, Part Time, Borrowed Time: A Note on Resources

Resources for your FinOps team ultimately should end up as fully dedicated to FinOps; however, they often start as a dotted line with a percentage allocation of time. A combination of the two models is also common: you have a dedicated FinOps lead who has a dotted reporting line to finance or engineering partners.

In the beginning, don't be afraid to consider external contractors or consultants to kick-start, accelerate, or even run parts of your central practice. It's very hard to hire good talent who knows FinOps well; sometimes the consultants available to you are the best option. The FinOps Foundation has a list of FinOps Certified Service Providers.

Often, a nondedicated virtual FinOps team can get the company started on FinOps. Use one to start to build some capability such as the basics of understanding your cloud spend and identifying the biggest cost drivers. As you get good at that, your success becomes your pitch for funding dedicated resources. Start small, look for wins, promote them widely.

As Mike found out at Atlassian, they knew there was value in optimizing spend, and that there was a need to understand costs, but it wasn't until he was able to show just how large the impact could be that he got buy-in for scaling a dedicated team. This came on the back of hard data based on real-world examples, not projections. The beauty of FinOps headcount is that it usually pays for itself. It's rare that you can say to your executive, "Give me five resources, and I'll give you the money back."

But what happens if your pitch gets knocked back and you don't get funding for the team? It's time to look at why the business is making that choice. It could be what we call *deep pockets syndrome*, where the priority is to migrate at all costs, and the executives want the teams to worry about getting it done above all else. In that

case, you may need to start practicing *informed ignoring* until the business is ready and begin giving visibility to the various stakeholders. Don't try to force engineers to do cost optimization. Start building your muscles around cost allocation, begin exploring the data with cloud native tools, and figure out what basic first capabilities you can start without the dedicated team.

Some early wins may be to begin exploring commitment-based discounts, as discussed in Chapter 16, to achieve some quick cost optimization gains. These discount vehicles can largely be made in a manner transparent to engineering teams, keeping them focused on higher-priority efforts.

A Complex System Designed from Scratch Never Works

Before building your pitch, consider the following quote:

> *A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over with a simple working system.*
>
> —John Gall, *Systemantics: How Systems Work and Especially How They Fail* (Quadrangle, 1977)

If you make your plan overly complex then it is unlikely to succeed. Your adopting FinOps plan should simply outline the steps the business will take to start—or expand—FinOps and the benefits of getting the practice going.

Keep it simple. Don't boil the ocean. Pick the most critical parts and start there. Define who the key stakeholders will be, determine what skill sets you already have and what gaps exist that need to be filled, define achievable objectives, and begin to lay out a cost allocation strategy.

You also must paint the picture that effecting real cultural change in the organization is a long process that can take years. FinOps needs to grow in stages and mature over time as the adjoining teams and processes recognize the changes required to effectively manage the value of cloud.

Like the human immune system, a FinOps practice is not born fully formed. To be ready to respond to challenges, it must start as a basic system that learns through experience the patterns, habits, and appropriate responses it needs to adapt to an ever-changing landscape.

Conclusion

Getting organizational adoption of FinOps takes careful planning, an understanding of executive motivation, and a lot of research into how the organization functions.

To summarize:

- The pitch for the advanced practice should include an assessment of the current practice and a plan for FinOps within your organization that includes the additional investments that will accelerate specific capabilities and their resulting outcomes.
- Don't try to overshoot with your initial plan for the FinOps practice. Start small, gain wins, and iterate over time.
- Different executives will look for different outcomes from the FinOps practice; be sure to understand their differing motivations before pitching each.
- Your pitch for starting a FinOps practice should focus on the low-hanging fruit that can be achieved with careful investment.
- Remember to overcommunicate and socialize your efforts at all stages to bring along the various stakeholders.

Remember, building a FinOps practice is about building consensus. You've got to win hearts and minds, and change long-standing processes. In the next chapter, we'll get into the FinOps Framework, which will give you a menu of capabilities to choose as you take your practice to the next level, from wherever you may be starting.

CHAPTER 7

The FinOps Foundation Framework

The FinOps Foundation Framework is a constantly evolving set of building blocks that enable you to construct a FinOps practice to match your organization's cloud maturity and complexity. This chapter introduces you to what the Framework is and how to use it in your FinOps practice. In addition, the chapter explains the Framework's structure and illustrates how you can adapt it to meet your organization's FinOps needs with related implementations, such as guides, assessments, and playbooks.

Adopting the Framework into your practice keeps you from reinventing the wheel and enables you to quickly remix known building blocks that have existed in various forms for years. Some parts of the Framework are relevant to all practices, such as cost allocation (i.e., metadata and hierarchy) and establishing a FinOps culture, and some are relevant only to specific scenarios, like managing shared costs or IT asset management integration.

The Framework outlines the principles, personas, maturity characteristics, and key capabilities to structure the activities and measures of success needed for a successful FinOps practice. The FinOps Foundation practitioner community, the cloud service providers, and partner members have all contributed to the FinOps Framework. You can relate everything in this book back to the Framework.

Every organization adopting the cloud will need to develop mechanisms to manage cloud costs. The Framework provides an operating model for your FinOps practice by aligning with cloud cost management best practices.

Rather than discover new, unknown pitfalls as you mature, you can use the Framework to structure your FinOps practice and maintain focus on all the areas of FinOps that help you achieve your goals earlier, while being aware of the elements you are not currently focusing on.

The Framework also serves as a mechanism for better communication between all the people in your organization assigned with FinOps responsibilities. The Framework forms the basis for implementations like the FinOps Assessment, which helps to evaluate and align key stakeholders around a common evaluation of your organization's maturity.

An Operating Model for Your Practice

Think of the Framework as an operating model for your FinOps practice. It is not a specific list of tasks to be completed in a specific order, since no FinOps practice is exactly like any other; rather, it is a comprehensive list of activities you may call upon to use in your practice as needed.

Imagine for a moment that you are planting a garden. Every day you assess what you see and decide what task will provide the most value. You might plant, prune, water, fertilize, mulch, and so on. There are plenty of guides to help you discover and develop the list of skills you need as a gardener. Your garden is growing and has ever-changing needs. You cannot simply mulch your way to the world's best garden. Following the same list of tasks every day won't allow you to adjust priority toward whatever might best benefit the garden at that specific time. You need to develop an *operating model* and a feedback loop that tunes your daily activities to the needs of the garden, adding and maturing capabilities based on the plants you have, the condition of the garden, the type of climate you are in, the amount of seasonal rainfall, changes to soil conditions, and other changing factors.

Similarly, in your FinOps practice there is no simple list of tasks for you to follow that will always lead to positive results. The Framework is there to help you understand the list of capabilities and the basic structure of how these capabilities are implemented within different organizations, and to allow you to select the capabilities and implementation details that suit your organization at that time.

The Framework Model

Figure 7-1 shows an overview of the primary components of the FinOps Foundation Framework. This section talks briefly about each of the supporting elements in the Framework: principles, personas, maturity, and phases. Then the next section describes domains and capabilities that form the main components of the Framework.

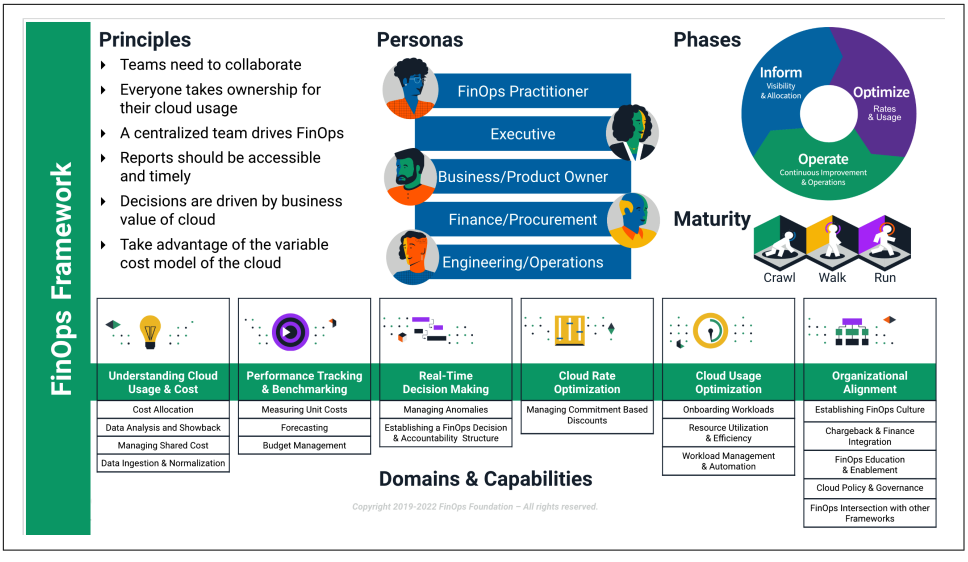


Figure 7-1. The FinOps Foundation Framework poster 2022 available online

Principles

The Framework includes the FinOps principles, covered in Chapter 1. All of the activities you perform in your FinOps practice must be consistent with the principles. The domains, capabilities, personas, and phases all reinforce the values they espouse. As you extend the FinOps Framework and customize its components for your organization, always keep the principles in mind and maintain consistency with them in your practice.

Personas

Although the FinOps practitioner needs to be involved in all aspects of the Framework, all personas involved in FinOps will also benefit from understanding its structure. Business leaders will be interested in understanding the domain outcomes and which areas to invest in to ensure their questions are answered and objectives are met. Engineering teams need to know how each of their tasks contributes to a particular capability and the Framework gives context to the outcomes of their efforts. A consistent language of FinOps is enabled via the Framework terminology and helps each persona develop their understanding of the role they need to play.

Maturity

The Framework applies to all practices regardless of the level of maturity. As you mature, you will adopt more of the capabilities over time and then set ever-increasing targets on the measures of success. Each capability itself will be at different stages of maturity.

Each capability in the Framework has a maturity assessment. You should aim to be familiar with both the capabilities you plan to tackle immediately as well as those you plan to delay starting. The list of capabilities is not a rote list you should aim to follow blindly. Maturing a capability takes time and continuous iteration as your organization grows into the activities required. The goal is not to be at an advanced maturity level across every capability; it is to be at the level of maturity in each capability appropriate for the needs of your organization. You should assess which of the domains will help you achieve the desired outcomes, then then which capabilities within the domains require improvement.

Shared costs are a common example of a capability that can be given too much time early on. These often occur when a single cloud resource or service is used by multiple teams. While you can introduce detailed sharing models via your own data crunching or tooling to reallocate costs according to real-time usage, such specificity is not always needed. While the tooling and best practices in this space have improved greatly over the past few years, you should assess whether having an agreed-upon fixed percentage—like splitting some shared costs based on an agreed percentage—is adequate for now. Simple allocation solutions can often meet the business needs early on without adding in the complexity of a more mature and dynamic shared cost capability.

Phases

Chapter 9 covers the FinOps lifecycle and the three FinOps lifecycle phases: *inform*, *optimize*, and *operate*. The phases help you organize the activities of each capability in a consistent order. They remind you to always work in an iterative, cyclical fashion by making incremental improvements as you gain maturity.

Domains and Capabilities

The FinOps Framework leverages a collection of domains and capabilities to implement the concepts in the FinOps principles.

The high-level domains encompass a set of desired outcomes for the business. The domains and their related outcomes may change over time, so we recommend viewing the Framework at the FinOps Foundation website. Table 7-1 shows the domain list and primary outcome at the time of publication.

Table 7-1. Framework domains and their primary outcomes

Domain	Primary outcome
Understanding cloud usage and cost	Drives accountability and transparency
Performance tracking and benchmarking	Defines what good looks like
Real-time decision making	Enables a fast feedback loop for data-driven decisions on spend
Cloud usage optimization	Ensures efficient use of cloud resources, while reducing wasted spend
Cloud rate optimization	Uses the various commitment-based discount programs to reduce the amount you pay for consumed resources
Organizational alignment	Aligns your cloud usage with your organizational goals and structure

These outcomes help you to answer the real-world questions you will get from your organization, such as:

- What is this application's biggest cost driver? (understanding cloud usage and cost)
- How can you reduce spending on a particular service? (usage optimization, rate optimization)
- How has your usage of a category of resource changed over time? (performance tracking)

Each domain then identifies a set of capabilities that help solve specific challenges you will face when trying to achieve your required outcomes. Each capability describes the required data, actions, and metrics that contribute to the success of the capability.

Structure of a Domain

The core sections of each domain follow a standard structure:

Definition
: Helps you understand the high-level activities you need to perform in your FinOps practice to achieve the desired outcomes you should expect and the primary questions that the domain aims to address for you.

Capabilities
: Provides a list of capabilities to achieve the outcomes for this domain. We'll talk about capabilities in more detail later in the chapter.

Vendors and service providers
: A list of tools and service providers that provide solutions to help you within this domain. Some solutions providers specialize in tooling that targets a single domain with deep expertise, and others offer holistic solutions across multiple domains.

Training and courses
: A list of the training supplied by the FinOps Foundation and certified training partners to help educate you and your staff in this domain.

Some domains may have additional content specific to the domain.

Structure of Capabilities

Each capability represents functional areas of activity supporting its corresponding FinOps domain(s). The core sections of each capability follow a standard structure.

Definition

This definition outlines the scope of the capability, which helps you understand which capability aligns with a specific goal or outcome.

Maturity assessment

The maturity assessment gives examples of the capability at various stages of maturity, which you can use to assess where you stand today and where to focus your work next.

Assessment lenses

By looking at your FinOps practice through a variety of lenses, you can assess the strengths of each with respect to your broader organization as well as specific target groups within it. As outlined by Ben de Mora from the FinOps Foundation, there are five lenses through which to assess your capability maturity:

Knowledge
: Does your target group understand what the capability involves? How broadly is this concept and its mechanisms, terms, and processes known?

Process
: Can your target group describe a process for conducting the capability in their context? Consider the efficacy, validity, and prevalence of such processes.

Metrics
: Is this capability measured? Do you have a way to prove progress over time? How are you obtaining those measurements, and how relevant are they to business outcomes?

Adoption
: How broadly has the capability been adopted and accepted by your business as part of its integral and critical functions? Consider the prevalence and presence of this capability across your entire organization.

Automation
: Is your target group automating this capability in ways that are repeatable, consistent, efficient, and labor-saving?

Functional activities

Functional activities are tasks or processes that allow various personas to meet the demands of a FinOps practice iteratively through the phases of the FinOps lifecycle (covered later in this book). These activities and procedures serve one or more of the following:

Enablement
: Building capabilities to help others perform a FinOps activity

Education/knowledge sharing
: Teaching others how to perform a FinOps activity

Advocacy
: Helping others understand the importance of the capability

Actionable tasks
: Creating and communicating a task that requires someone in your organization to take action

Improving FinOps maturity
: Developing processes and tooling that enable FinOps to become more successful within your business

The number of functional activities implemented within an active FinOps practice will increase as the maturity of FinOps increases.

Measures of success

Each capability needs to have at least one metric that monitors its performance and allows the FinOps practitioner to set targets. These metrics and the associated targets measure how successful you are in a given capability. As your practice matures, you will increase targets, allowing for short-term goals to be set and objectively measured. Measures of success can also help you identify which activity is holding you back from achieving your goals.

Some examples of measures of success in various capabilities include:

Cost allocation
: Percent of cloud spend that can be allocated to its owner

Forecasting variance
: Percent of variance between forecasted spend and actual spend

Resource utilization
: Percentage threshold of monthly cloud spend that is driven by idle or inefficient resources

Inputs

Inputs that can help with the success of a capability include external data sources, reports, whitepapers, and training. Some examples are cloud adoption frameworks, vendor recommendations for optimization, and business policies/standards. This list of resources can be a great way to get further educated or begin collecting the information needed to set up functional activities.

Adapting the Framework to Fit Your Needs

Because different organizations have different operational needs that shape their path to success, the FinOps Framework itself is deliberately designed to be directional by providing parameters for success through a range of KPIs, by describing maturity characteristics in the context of required activities, and by describing the responsibility of each FinOps persona. For example, a public sector or government agency will have a very different budgeting and forecasting process than that of a private company. The Framework is expected to be adjusted and updated to fit your organization's operational needs.

Which cloud service providers and vendor tools you are using will further affect how your teams complete specific activities within your FinOps practice as described by the Framework. The recommended approach is to start with the FinOps Framework and modify the functional activities to capture any organizational or industry-specific activities. Examples of this include change management workflows or purchase approval processes. For each activity, assign them to a specific persona within your organization, educate staff on the process to be followed, and advocate the importance of completing each activity.

Framework implementations are resources created by the community to provide industry- and vendor-specific recipes for success. These resources come in many forms, including how-to guides, capability playbooks, and even video presentations. Framework implementations are contributed by individual members of the community, community working groups, and partner members. The aim for implementations is to provide contextually relevant approaches for tackling challenges to improve your FinOps capabilities. Implementations can describe how to use specific tools, processes, or refined functional activities for an individual capability, domain, or the entire FinOps Framework.

Stories from the Cloud—FinOps Community

A cross section of US government agencies came together in early 2022 to create the first government playbook for how to do FinOps within the unique constraints of government procurement processes and rules. Nearly two dozen agencies participated, including the Army, Federal Aviation Administration, White House, Small Business Administration, Department of Housing and Urban Development, General Services Administration, and Department of the Interior. They were led by Melvin Brown II, Deputy CIO of the Office of Personnel Management (OPM).

As they state in the first version of their implementation guide:

> This playbook is designed to build upon the existing FinOps framework to assist federal agencies as they begin (or mature) their cloud journey. While each agency should tailor their FinOps implementation to their organization's current state and desired outcomes, this playbook highlights areas within the FinOps framework which will likely be slightly tailored for a public sector use case.
>
> This playbook will help guide federal agencies towards a successful cloud financial management practice, enabling key outcomes afforded by near real-time cloud spend visibility and reporting of executive level details to align an agency's cloud operations with strategic goals.
>
> The playbook is organized into three stages which are designed to assist in setting up your agency in implementing FinOps, allowing you to smoothly transition into operating FinOps within your organization.
>
> As you develop your own solutions, you can contribute those back to the community as implementation guides that will help others following after you who find themselves in a similar situation.

Implementation guides are a great reference to develop your FinOps practice by seeing how other organizations have implemented individual capabilities or even the entire Framework. If you cannot find an example that suits your goals, we recommend you join the FinOps Foundation to use the community of practitioners there to your advantage. They are always willing to answer questions, share their insights, and demonstrate solutions that worked for them.

Connection to Other Frameworks/Models

The existence of other financial models and frameworks, such as IT Financial Management, Technology Business Management (TBM), IT Asset Management (ITAM), or IT Service Management (ITSM), can also impact which personas, specific processes, and functional activities are implemented. Many successful FinOps teams work alongside these existing operational models within their organizations. There

is little value in having FinOps replicate or replace existing capabilities in existing operational models that are functioning well. FinOps can plug the cloud financial management challenges these different operational models have and integrate them to combine areas of strength into one holistic system. The FinOps Framework has specific capabilities to share outcomes with these other practices.

For FinOps to work well alongside other operational models/frameworks, you need to identify the functional activities that FinOps will be responsible for and those of the other model. In addition, you'll need to track which information you shared between the two to keep consistency across them. This information is covered in greater detail in Chapter 25.

Conclusion

The FinOps Framework is a valuable resource not only for those practitioners starting a FinOps practice but also for mature practitioners.

To summarize:

- The FinOps Framework is a set of building blocks to allow you to create a successful FinOps practice that matches your organizational maturity in the cloud.
- Adopting the terminology used in the Framework will help with a common language used by all personas about FinOps within your business.
- The Framework enables you to assess where you have strengths and weaknesses.
- The Framework is not only for the FinOps practitioner but also for all personas of your organization.
- You should apply the Framework to your own practice by adapting the capabilities to fit into the way your organization operates.

The next chapter looks at considerations for building reports and dashboards that make up the *user interface* (UI) of FinOps.

CHAPTER 8

The UI of FinOps

Your reports and dashboards provide the user interface (UI) for FinOps to your organization. By looking at your reports as your UI, you can apply decades of research into effective UI design. You should be looking carefully at those reports as the *product* of a FinOps practice. As such, it's the interface through which your users interact with the data. How you present FinOps data in reports influences the success of FinOps enablement in your organization. The FinOps product should be treated as a production workload: quality is important, clarity is important, and user experience (UX) is important. These lead to trust in the data, which enables decision making by teams and prevents time wasted defending the quality of the data.

It's easy to make reports containing pretty graphs and tables of data, but it's harder to tell a story to your users based on what they need to know and enable them to make decisions about what actions they need to take. This chapter covers some important concepts to help you build high-quality FinOps reports. We'll use the word *report* for the rest of this chapter, but these concepts and suggestions apply equally to dashboards and business intelligence systems used to deliver FinOps data to your users.

Build Versus Buy Versus Native

When thinking about the reporting you'll create as part of your FinOps practice, first consider the tooling and platforms you will need to do the job. As you learned in Chapter 5, the volume of data you must process on a continuing basis to create FinOps reporting can be huge, so you will likely have a whole toolbox of tools. The question is what type of tools you will use. The concepts in this chapter are not meant to imply that you should build your own FinOps reporting platform. They also aren't meant to discourage you from doing so. We commonly see multiple tooling paths to successful FinOps data management:

- *Native tooling* such as AWS Cost Explorer, Google Cloud Cost Management, or Azure Cost Management are always the place to start when beginning your FinOps journey.
- *Software-as-a-service (SaaS) platforms* like the Certified FinOps Platforms designated on *FinOps.org* are often procured to go beyond native tooling and speed time to value in relation to rolling your own systems. They also act as a critical abstraction layer between you and the regularly changing landscape of multicloud billing data.
- *Buy then augment* is an increasingly common theme we're seeing where large cloud spenders will take the APIs of either native tools or platforms and add custom functionality around the edges of their scopes as well as integrate data into and out of them.
- *Homegrown* tooling was the fastest-growing segment in the 2022 State of FinOps data; sometimes built from scratch but often built on top of existing business intelligence (BI) tooling (e.g., Tableau or Looker) or using cloud-specific solutions (e.g., AWS Cost Intelligence Dashboards or Google's BigQuery Export).

As with many things, the answer to which approach you take depends on your company's resources, culture, and skills. We know some successful large FinOps practices that rely entirely on a platform and some that rely entirely on homegrown tooling. Most, if not all, organizations will rely upon a collection of tools and approaches, which can change as their maturity grows and their cloud use changes. There is no right answer.

When to Use Native Tooling

Each of the cloud providers offers its own set of cost management tooling. We recommend you always start with the native tooling. It typically provides the core capabilities you need to cover at the earlier stages of maturity. As your cloud spend and complexity increase, it's highly likely that you will need more than native tooling can offer. At that point, the conversation becomes whether to build or buy. Whichever path you take as you advance, native tooling still serves as a sanity check for the data coming out of your platform, and it typically will be the way you communicate with your cloud service provider's support teams. Stay up to date on the native tools and ensure the right people in your organization have access to them.

Typically, native tooling falls into two categories. First are purpose-built cost management tooling such as AWS Cost Explorer, Google Cloud Cost Management, and Azure Cost Management. Secondly, as you mature, there are also BI tools—such as Google's Looker and AWS's QuickSight—with report templates provided by the cloud service provider that enable you to more deeply customize your reporting through tools.

When to Build

On the FinOpsPod podcast, Kim Wier, Director of Efficiency Engineering at Target, tells the story that her CIO sees Target as a product company, and as such most of their development happens internally:

> We need to have the knowledge in-house at Target. Our CIO did not want to outsource the technical knowledge to other companies. Target's development by and large happens in-house. We've built up our software engineering culture, and so we do all that development internally.

As a result, Target invested in a strong FinOps engineering team to manage the data pipelines, ingestion, as well as frontend development teams working on developing custom dashboards for the various personas, like engineers and leadership.

Here are some common reasons people choose to build their own reporting layer:

- Control over your own data
- Easily integrate with internal data (unit economic data, taxonomy data, performance data, and so on)
- Customize capabilities based on specific company needs
- SaaS models often don't have required compliance (e.g., FedRAMP, SOC 2)
- Fee structures don't always scale with value
- Flexibility in reporting
- Ability to quickly make changes
- Technical ability to perform complex software development and data analysis within the company
- Organizational buy-in to continuously invest in learning cloud data details, which are constantly evolving and expanding

Stories from the Cloud—FinOps Community

This story from the FinOps Foundation member Slack forum was shared by Lindbergh Matillano, Director of Cost Optimization at Avalara:

> I follow the philosophy of People, Process, and Tools. The right people need to be in place first, followed by processes, then tools. I often see organizations invest in tools before they have the right people and processes in place. This often leads to underutilization of the tool or ineffective/inefficient processes designed around the tool.

I've started/run FinOps practices for at least four companies. When I enter an organization, the first thing I do is run a usage report to answer the following questions:

- How many people have accessed the platform in the past three months?
- What reports are regularly used?
- Which features are being used?

If there are low utilizations, I dig into the reasons for the underlying cultural reasons which are driving this low usage when evaluating the ROI of a platform.

What I often find are stakeholders:

- Prefer native tools (e.g., Cost Explorer)
- Request modifications of standard reports that are outside the capability of a platform
- Only use 10% of a platform's capability (mostly reporting related)
- Want to understand the code of a platform's automations so they don't allow automatic remediation, especially in production instances
- Trying to tailor a platform to the specific needs of an organization often requires customization or workarounds adding cost to the tool.

For a build or buy platform to be successful it needs to be supported by a culture of cost awareness, as tools fail without an underlying FinOps cultural focus on costs and the value of cloud.

In some cases, buying a platform saves development time and resources. But I think getting engineers involved in building solutions helps embed a culture of cost awareness.

Why to Buy

Conversely, RJ Hazra, CFO of Global Technology at Equifax, chose to buy a platform to more quickly move toward focusing on the cultural aspects of FinOps, but also to focus on automation efforts:

> We used a procured FinOps platform to generate visibility. But it wasn't enough—we needed to be able to quickly take action on the visibility. Finance and engineering needed to partner more closely than ever. The CTO prioritized a FinOps cultural transformation and empowered SREs [Site Reliability Engineers] in the organization to help enable and convince other engineers. We needed an engineer talking to an engineer on usage optimizations rather than building tooling. We needed to figure out how to structure the commitments of the cloud providers so we also brought sourcing/procurement into our team.

Michael Barba, Senior Cloud FinOps Manager at Couchbase, offered his perspective on the *why buy* a FinOps platform:

> A lot of companies probably don't have the engineering resources in-house to build out homegrown tools, or if they do, those individuals are more valuable elsewhere. For Couchbase, that meant focusing on building out the services we sell instead of building internal reporting that we can outsource via software.

Here are some common reasons people choose to buy an SaaS platform:

- Fast time to value: usually very little IAM and cloud console work to set up
- Simple tools to create custom groupings that map spend
- Complex math like amortized RI mapping is already calculated
- More detailed optimization features than what the native tools offer
- Access to subject matter experts at platform companies
- Platform providers mitigate risk of changes in cloud data and data quality

Stories from the Cloud—FinOps Community

This story from the FinOps Foundation member Slack forum was shared by Angel Alves from Saint-Gobain, a French multinational manufacturing company:

> When I joined my previous company they were already using a 3rd party tool. I tried to revert back to a self-built solution but the challenge was a lack of resources. I could replace like-for-like reporting (single vendor, no Kubernetes), but our roadmap had adoption of another cloud provider plus Kubernetes. When making the effort involved to build a solution to support that roadmap, it was decided that there were other priorities for the engineering resources, so we chose to continue paying for the 3rd party platform, and instead focus on facilitating the adoption of the new cloud provider and Kubernetes technologies.

Another person in the Slack chimed in with their view related to platforms:

> For us, it's simply that we don't have the resources/time to build it ourselves. Using a 3rd party platform saves us so much time and allows us to keep up-to-date automatically (new AWS services, Kubernetes support, etc.). Although it's improving, native tooling isn't good enough yet, at least for now. We often see people asking questions about "how can I do XYZ" and it's usually already offered by the third-party tool we use.

We encourage you to look at the concepts in this chapter as lenses through which to make any of the paths we've discussed more successful. FinOps relies on driving accountability and decision making, which makes it inherently a UX challenge all the way from conversations with colleagues to the data you present in reports.

Operationalized Reporting

Your reports and dashboards must present information consistently, with high data quality and with a high degree of review and control. You should treat your FinOps reports the same way your engineering teams treat their production services, with controls, quality checks, and rigorous review. If your reports break or start presenting incorrect or even inconsistent information, then FinOps inside your organization is adversely impacted.

Data Quality

Mike's colleague Diana Mileva, a Senior Business Analyst of Cloud FinOps at Atlassian, presented at FinOps X 2022 about data quality within FinOps reports. Diana explained the importance of data quality and different approaches on how FinOps teams handle data issues. The key items in Diana's presentation included data delays, data issues, quality checks, and reporting challenges FinOps practitioners face.

> *When someone questions our FinOps data, we treat it like a production bug and open a Jira ticket to investigate. We perform automated data quality checks across the entire data set along with SLOs [service level objectives] for timeliness and accuracy. We aim to detect any issues before our users and inform them proactively so they do not make decisions based on incorrect information.*
>
> —Diana Mileva, Senior Business Analyst at Atlassian

At the time of this book's printing, in most clouds there is a delay between using a cloud resource and receiving the billing data for its use. The delay tends to be 24–48 hours. Unfortunately, the delay is not consistent and can be longer, especially around the start and end of a month. You may sometimes receive partial data for the most recent period of billing data available. The impact is that FinOps reports can erroneously show a drop in costs during more recent hours of reporting and can be missing the last day or two of data—see Figure 8-1.

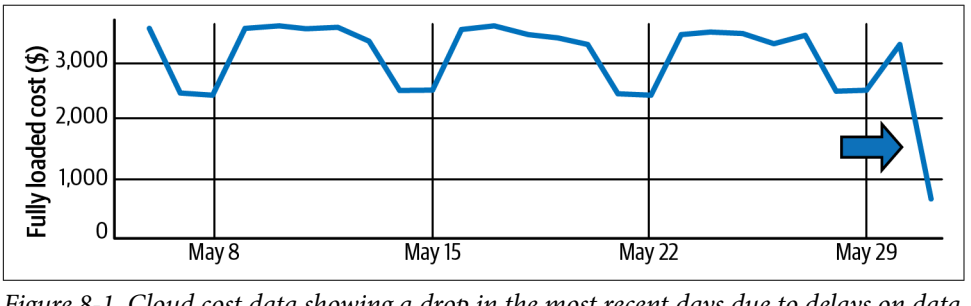


Figure 8-1. Cloud cost data showing a drop in the most recent days due to delays on data delivery

A common approach is to remove the latest couple of days of data from reports to avoid this drop in costs. Although trimming the available data is very helpful in making the data presented more consistent, it's not a silver bullet that removes the need for any additional monitoring and alerts on the quality of the data reported.

For advanced practitioners building their own reports, we recommend you display two dates on your reports, with one date representing when a report was last updated, and the other indicating how recent the data available was when creating the report. This is especially important on reports that do not show an hourly or daily breakdown and offer only summarized data.

There have been numerous occasions where cloud service providers have had issues with delivering customer billing data that result in incomplete or inaccurate data for a period of time before it is corrected in the finalized bill after month end. These data issues often cause problems—using either in-house tools or vendor-provided solutions—with enriching the cloud service provider billing data in post-processing.

To detect issues with billing data, it is important that you monitor the quality of the data in your billing reports. This allows you to alert your staff when the data presented in reports has an issue. Being proactive on alerting staff helps you maintain trust in the FinOps reports. The cloud service provider can change the way data is reported. While the top cloud providers offer more details about their report structure today and often publish information about coming changes, it's not always the case. We have seen cloud service providers fix inconsistencies in the billing data that can lead to breaking the methods a company had in place to handle the inconsistent data.

FinOps reports are built upon queries, which can incorrectly handle specific events. Examples include the change of year, where a query was incorrectly written to handle the date rolling into the next year, and queries that assume specific values are present, which filters out data accidentally. It is important for you to validate your queries not just when you first build your reports but also over the time your reports are available. This can be especially important when taking reports created by others either internally to your company or externally from the community. Knowing the limitations of the queries used underlying these reports can avoid a possible situation where the information that is presented is misunderstood or incorrect.

If your users find the data presented in your reports to be incorrect or inconsistent, then trust in your reports erodes. This loss of trust can lead to inaction from engineers or the proliferation of unsanctioned tools and reports that are often inconsistent with your official FinOps reports, resulting in a lack of decisions being made or decisions made with incorrect data. Further, inconsistent reporting will lead to large amounts of time spent reconciling the differences and discussions about correctness

of the data. The point of the FinOps team providing a consistent UI into the cloud cost data is to save the engineering teams the time and effort of teasing out these details themselves. When they don't trust the reporting, not only can cost savings disappear but valuable engineering hours are also lost.

As Diana Mileva went on to say in her 2022 FinOps X talk, "Individual reports of data quality are treated like production incidents to be investigated quickly, including consistent communications back to the reporter and a postmortem of any actual issues. Through this process we continually improve the quality of our data and we also build trust with our stakeholders that we take data quality issues seriously."

Preventing bad data from getting into your reports is important, but sometimes this can be impossible to avoid, so building processes that detect data problems and proactively alerting your users can help maintain this confidence in your reports.

Perfect Is the Enemy of Good

Data quality does not mean 100% real-time accuracy to the micro-penny. It means data that is consistently good enough to make informed decisions. It is inevitable that some errors or inaccuracies will creep into your reports for one or more of the issues discussed. But the job of the FinOps team is to be taking steps—commercially reasonable efforts—to ensure that the data presented is consistent, current, and has enough accuracy to enable the right decisions for the organization. Over time, adjustments can be made to the timeliness of processing available information, data ingestion and normalization processes, added data sources, and query refinement.

> *Above all else show the data.*
> —Edward R. Tufte

Report Tiering

The reports you build to surface data for your teams will not all be equally useful or official. There are reports you rely upon to represent official chargeback reporting, and there are other reports that simply answer an ad hoc question. Report tiering can be an effective way to differentiate reports.

Diana goes on to share: "Internally we mark our reports as *managed* or *unmanaged*. If you want something built ad hoc, we'll create an unmanaged report and mark it as such. There becomes a long list of unmanaged reports that are not kept to date. For example, the queries may need to be updated as new data schemas come into the billing data, or the query was written in a way that was specific for that month. In order to keep trust in the reporting, be sure to maintain and indicate which are your managed reports and those that are not maintained."

A mature FinOps practice may accumulate hundreds of FinOps reports over time, many of which were created at a specific point in time to answer a question the

business had previously asked. Ideally, the team is continuously curating a much smaller and focused set of primary reports that represent the gold standard for what it believes distributed teams should be paying attention to the most.

Stories from the Cloud—Jason Rhoades

Jason Rhoades, Development Manager at Intuit, shares:

> We have fewer than ten official reports which answer all the questions for which a graphical report is the right way of answering. The data in those reports is narrowed to the portion of the cloud of interest to a party via filters.

He went on to note that data democratization can be helpful, with some caveats:

> You don't want to be a silo closed off to everyone; at the same time, opening up comes with the risk the data consumer will make mistakes—or even just different decisions—in how they build their custom products, such that your FinOps team gets entangled having to explain why the numbers in Team X's custom report differ from the mainstream report.
>
> This is definitely a risk, but should be weighed against the idea these groups may discover or prove the value of a new perspective or capability the core team would not have thought of, or had the cycles to develop themselves. Effectively crowdsourcing FinOps innovation.
>
> Clear boundaries of responsibility should be established when the FinOps team becomes a data provider to limit its liability in the event that data is misused, or rather misunderstood by the consuming team.

With this growing collection of reports, it becomes unclear to your users which reports are important, which are up to date, and whether the user can trust the data. By clearly marking your reports, you can guide your users to know which is which.

Ideally, you apply a tiering or class system to your reports, from low tier reports that are marked with a creation date and labeled as nonmaintained, all the way through to high tier production reports that your users can easily identify, signaling that they are well maintained, monitored for quality, and that your users can trust them.

Ad hoc reports made for a specific purpose to answer a specific question at a point in time are an important part of the FinOps process. However, you should aim to constantly remove old reports that are no longer relevant or providing value. Doing so makes it less likely that your staff will find the wrong report and use it to make decisions.

Give a staff member the link to a specific report, and they will be back next week asking for the link to another. Provide your staff member with the link to an updated list of your production reports, and they will be set forever. This is obviously a play on a common proverb related to fishing, but it's very useful in thinking about how

you guide your staff to the FinOps data you make available. Although it's helpful to guide them directly to the report that helps them with the current situation, making sure they also know how to find your reports in the future will reduce the support burden on your FinOps team.

Rolling Out Changes

Over time, your reports will need to change: new data will become available, and some data will become irrelevant. The next section covers the universal report, but sometimes your reports might need to be divided into two. Changing your production reports without notice will impact your FinOps practice, as your users will not understand why the changes have been made, what the new data represents, and where they can go to get access to the data that was taken away.

Manually changing your production reports runs the risk of leaving the report in a broken state, with the result that your users will consider the report untrustworthy and avoid the data going forward. Having a staging copy of your production reports allows you to test the changes you intend to make and provides you with a location to build some documentation for your users to read before the changes are made to your production reports. Bringing your users along with the changes as they are implemented will ensure they understand how to continue to use your reports.

Ideally, your FinOps reports can be configured using an API or saved and restored using some sort of file upload. This provides you with the ability to store the configurations of your reports in a configuration management system or code repository. Keep the previous configurations of your reports to enable you to quickly restore your reports to a known good state if a change does not go to plan.

The Universal Report

As the number of reports grows, there is a tendency for practitioners to try and consolidate all FinOps data into what we call the *universal report*—one report to rule them all. This universal report is overloaded with all the data you can fit, without consideration to the message it presents or how clearly the data is being presented to your users. The universal report is almost always hard to read, and this detracts from your users' ability to understand the information being conveyed. These reports often have no structure or organization to the data and just provide numerous pieces of information without any connection between them.

Consolidating reports together is acceptable when there is a clear connection between the data points and a story of how your users will read and understand the report. On the other hand, you should review your reports and identify when it would be better for your reports to be broken into two or more clearer reports, each easier for your users to understand.

When designing a new FinOps report, try to keep it simple. Include only the required information, summarized as much as is practical. You may have hundreds of columns of data that you could add, but including too much will make the report harder to understand. Or, even better, provide interactive reporting that allows users to drill down on information they need, add more context, or connect more data points only when it's required for them individually.

This becomes especially important as you build reports that are delivered to your users versus reports that your users load when they need the data. Delivered reports will arrive without context and need to be short and clear as to the message they provide. If a delivered report is complex and has too much information, your users will file the report away for when they have more time to read it, which of course they never do.

> *Clutter and confusion are failures of design, not attributes of information.*
> —Edward Tufte

With the ability to set up reports that will alert you to changes, you can create reports that monitor for things you do not expect to see, such as costs from cloud products you don't use, unattached storage volumes, or daily cost jumps of more than a set percentage. The notification function of these reports means you do not need to check on their status daily. Instead, allow the report to do the work of letting you know when you need to read it.

FinOps reports are most successful when you identify the common questions that your staff need to answer and target the report to answer these questions. Ask yourself, "What question am I trying to answer with this report?" and make sure the report achieves this goal. Also, consider the information your staff need to know to use your report. For example, if they need to have an application name or cost center number, ensure that you direct your users to reports that help them get that information first.

Think of your reports as layer cake with the topmost reports needing the least amount of information to use and providing only enough information for them to use the reports in the next layer down. As your users move down the hierarchy of reports, they are drilling down into more details of a specific area with finer granularity. Drilling down into the information is required only when your users need the extra detail to answer the questions they have.

Accessibility

Making your FinOps reports accessible to all your staff is essential. As you build your reports, consider that you will not be able to explain the report to all your users. If

you have to be there when your colleagues read your reports, then you will struggle to scale FinOps across your organization.

This section provides some common design guidelines used in UIs that will help your FinOps reports be more useful. If your organization is lucky enough to have professional reporting or UI/UX teams, they may also be able to provide you with examples in use already that can help further.

There have been so many great books in the domain of information design over the years. One we can recommend you read is *Envisioning Information* (Graphics Press, 1990) by Edward Tufte, who is quoted throughout the chapter.

Color

While drafting the first edition of this book, one mistake we made was to write content that referenced the colors within the figures we included. It turned out the book was going to be printed in black and white, with the figures now having various shades of gray and the content saying things like "the red line versus the blue." This removal of color made many of our figures unclear, and we had to redesign them and rewrite the references. Although achromatopsia—seeing in only black and white—is uncommon, having staff with some form of colorblindness is likely, especially in larger organizations. For this reason, you should aim to use more than just color to represent different elements of your reports. Instead of having just a red and blue line, make one solid and the other dashed. Instead of bar charts with solid colors, choose to have one with stripes and the other dots. If changing the format of the charts is not possible, it might be a better idea for you to have two charts versus one with multiple items stacked.

Visual Hierarchy

Lay out content in your reports and dashboards with related items close together. Make visual borders around related content and draw a clear delineation between indirectly connected content. Considering how you lay out content will make your reports more intuitive for your audience to read.

You should consider this visual hierarchy for the arrangement of information on reports, where summary information is generally at the top or left, and more detailed—or less important—information is below or to the right.

Usability and Consistency

Your users will quickly abandon reports and dashboards that are difficult and slow to interact with—adding to the challenge of implementing FinOps across your

organization. As you build a collection of reports and dashboards, aim to maintain performance and consistency, making it easier for your staff to understand how to use them.

Language

Chapter 4 covered the language of FinOps. Using the same terms in your reports when discussing FinOps will maintain consistency. Don't change terms between reports or use unclear terms that will cause your users to ask which meaning you intended. One inconsistency we often see FinOps practitioners make when creating reports is using the term *cost*. As we mentioned in Chapter 4, this could be anything from unblended costs or amortized costs or fully loaded costs.

Consistency of Color and Visual Representation

The need to use more than color as a differentiator doesn't preclude the use of color. But aim to be consistent with your color and visual representation use. When you present savings as a green dotted line in one chart and then use a solid blue line for savings in the next, it won't be clear to the reader that both represent the same thing. Arguably worse is using the same color/line style across charts for different items, forcing the user to carefully refer to each chart's legend to determine what is what. Inconsistency within the same report or across a suite of FinOps reports will trick your staff into misreading the information and erode trust over time.

Recognition Versus Recall

When data is presented together, a user can more easily recognize the connection between discrete pieces of information. If you present information referring to data that is either offscreen or in another report, you force your staff to recall the information previously presented, leading them to scroll around or switch between reports to remind themselves of the values. The separation of related information can make it harder for the reader to connect together. This connection is essential for long reports or large dashboards. You should try to keep related items together and not force the user to recall information presented elsewhere. If you do need to reference information presented elsewhere, you should either repeat the critical information or assist in linking them back to the correct location.

> *Comparisons must be enforced within the scope of the eyespan, a fundamental point occasionally forgotten in practice.*
>
> —Edward Tufte

You may find that the same top three summary boxes may appear on a lot of different dashboards you produce, because some summary-level information is always relevant

as a starting point, but then the tables or numbers that follow at lower tiers differ for different personas or business purposes.

Psychological Concepts

Dale Carnegie, author of the seminal book *How to Win Friends and Influence People* (Simon and Schuster), said, "Success in dealing with people depends on a sympathetic grasp of the other person's viewpoint." FinOps is largely about providing the right information in the right way at the right time to influence and encourage behavior by others.

Designing your reports for usability is important, but considering what information you include in your reports can also influence what your users take from them. The field of cognitive behavior or cognitive psychology has exploded in the last 20 years, teaching us more and more about the biases and behaviors that are common to us humans and how to apply techniques to take advantage of (or mitigate for) those biases.

To be clear, this isn't a chapter about how to trick people into doing things with cognitive behavioral science, but rather about understanding some of the biases people will have, because they're people, when reading your FinOps reports, and how to design around them or present information in ways that decision makers won't fall into well-known cognitive traps.

Let's see how a few psychological theories can be applied to make your FinOps reports more effective.

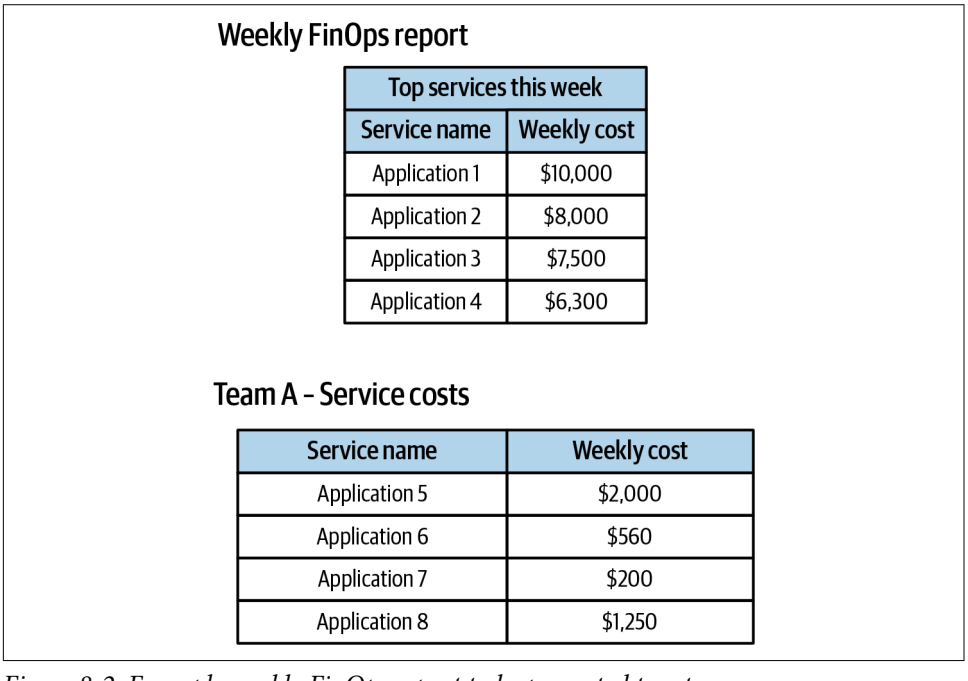
Anchoring Bias

A *cognitive bias* is often in play when someone has an error in their thinking about or interpreting information. *Anchoring bias*, or the *anchoring effect*, is a type of cognitive bias in which the information presented to the user early on affects how they perceive other information later. From Wikipedia:

> The *anchoring effect* is a cognitive bias whereby an individual's decisions are influenced by a particular reference point or 'anchor'. For example, an individual may be more likely to purchase a car if it is placed alongside a more expensive model (the anchor). Prices discussed in negotiations that are lower than the anchor may seem reasonable, perhaps even cheap to the buyer, even if said prices are still relatively higher than the actual market value of the car.

Your local retailer uses this all the time. Every time you enter a shop and see that expensive TV with an almost unrealistic price tag, this anchors your mind on what expensive is, so all the other TVs you are more likely to purchase appear to be a great deal in comparison.

Consider a report that presents your engineering teams with the costs of their services (see Figure 8-2). At the start of the report, you include the top 10 most expensive services in your entire organization, which anchors your engineers against the cost of these most costly services and could affect how they view their services.



Weekly FinOps report

Top services this week	
Service name	Weekly cost
Application 1	$10,000
Application 2	$8,000
Application 3	$7,500
Application 4	$6,300

Team A – Service costs

Service name	Weekly cost
Application 5	$2,000
Application 6	$560
Application 7	$200
Application 8	$1,250

Figure 8-2. Example weekly FinOps report to be presented to a team

Presenting the costs of their services alone, without the spending of other teams, will help avoid this bias and allow your teams to assess the cost of their service more fairly.

Although the decision to include this extra information may have been intended to be informative, for the desired outcome of the report, you should consider what information is required and evaluate the effect of adding in extra data points that may anchor your users.

Confirmation Bias

Another type of cognitive bias is confirmation bias. Users seek information from your reports and interpret it in a way that supports their prior beliefs. From Wikipedia:

> *Confirmation bias* is the tendency to search for, interpret, favor, and recall information in a way that confirms or supports one's prior beliefs or values. People display this bias when they select information that supports their views, ignoring contrary information, or when they interpret ambiguous evidence as supporting their existing attitudes.

Providing information across different reports allows users to select the ones that support their thinking. Consistency and recognition can help counteract confirmation bias.

Avoiding the universal report design can also thwart confirmation bias because it focuses every report or dashboard on a specific question rather than providing 10 or 20 data points from which the user can find *some* good news on which to focus attention.

As an example, having a report that includes all 40,000 compute resources that might be valuable to resize, grouped by application team, allows almost anyone to find someone who is doing worse than them. A different report entitled "Resources [Yourteam] Should Consider Rightsizing as Soon as Possible," with the top 5 or 10 money-saving resources, provides a list no one can argue with.

The Von Restorff Effect

Another psychological theory related to how people recall information is the *Von Restorff effect*, also known as the *isolation effect*. From Wikipedia: "The Von Restorff effect, also known as the 'isolation effect,' predicts that when multiple homogeneous stimuli are presented, the stimulus that differs from the rest is more likely to be remembered."

When presented with a group of items, the one that differs from the others is likely to be remembered. When applied to your FinOps reports, it's most likely that the biggest or smallest items are the ones your staff will remember (see Figure 8-3 for an example). Or, if one of your charts uses unique coloring or style, it will stand out more in memory than others.

Service name	Weekly cost	Difference from last week
Application 8	$1,250	+1,000
Application 6	$560	+$50
Application 5	$2,000	+$15
Application 7	$200	-$25

Figure 8-3. Service costs including weekly difference

If you present your engineers a report that shows the cost of their services and show only the current weekly spend of each service (as in Figure 8-2), then your engineers will remember the service with the highest cost. Instead, if you also include the amount each service has grown month on month, you can influence your engineers to recall the service that is growing most quickly. Using this effect, you can steer

engineers toward the outcomes you want reached when reading your reports, i.e., to focus on the line items with the most growth.

Hick's Law

To help your staff make a decision, your reports often present multiple options to decide upon. *Hick's law* is a simple idea that says that as the number of presented options increases, so too does decision-making time. From Wikipedia:

> Hick's law describes the time it takes for a person to make a decision as a result of the possible choices: increasing the number of choices will increase the decision time logarithmically. The Hick–Hyman law assesses cognitive information capacity in choice reaction experiments. The amount of time taken to process a certain amount of bits in the Hick–Hyman law is known as the "rate of gain of information."

Optimization reports in FinOps will often include a few alternative recommendations for your engineering teams to consider. You should aim to present only the best recommendations that are most likely to be selected.

Consider a recommendation that has multiple options.

Two of these options would result in only a small reduction in cloud costs, and another is a high-risk change for your engineers. Removing the items that are not effective reduces the total items for your engineers to consider from five to two and is likely to result in a quicker decision. In Figure 8-4, the AWS Compute Optimizer console limits the recommendations it presents to the user and shows the price difference and performance risk to help the engineer in making a decision. Keeping recommendations to a minimum has the added benefit of reducing the amount of time engineers have to spend reading all the options and can lead to higher productivity.

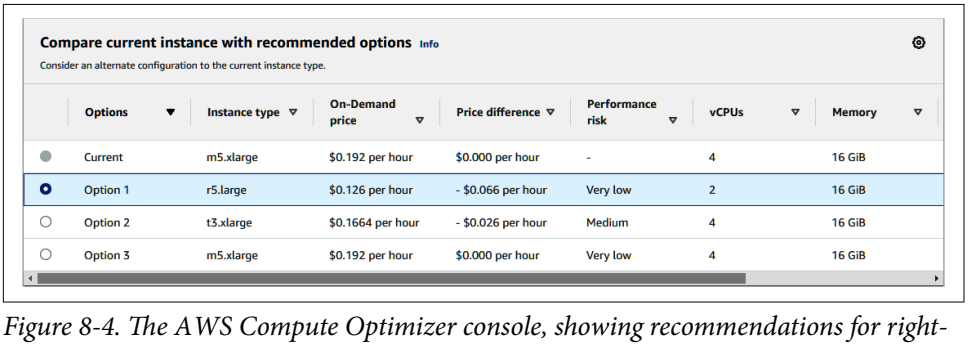


Figure 8-4. The AWS Compute Optimizer console, showing recommendations for right-sizing an EC2 instance

Hick's law and its implied limitations on choices ties nicely to the FinOps Foundation Framework capability: establish a FinOps decision and accountability structure. Policy decisions like what kinds of optimizations will be recommended (only those over a certain threshold of savings, only those on these types of resources, only those for teams that have adopted a certain level of efficiency, only those for resources below a minimum efficiency threshold, etc.) provide a model for minimizing the options your FinOps team presents and provide transparency for why you select or exclude certain ones.

Perspectives on Reports

When evaluating the design of your collection of FinOps reports, consider the perspectives discussed in this section. They will help inform the desired outcomes for your reports and influence the essential elements that you need to include.

Personas

In contrast to the universal report concept discussed earlier in this chapter, consider when it is better to have persona-focused reports that deliver the details needed for a specific role within your organization versus trying to make a single report that can answer all the questions for all the personas. For example, keep leadership reports—which roll up the specific details into a high-level overview of how FinOps is performing within your organization—discrete from the reports that engineers use to track their individual service costs. This allows you to build more targeted reporting that is most effective at driving FinOps outcomes. It is critical that these reports be consistent with each other in terms of metric definitions and language. If one of your reports is showing that a metric is off track, the other should reflect relevant opportunities for improvement of that metric.

Maturity

As your FinOps practice matures, the amount of information that can be reported will grow quickly. And as you implement additional capabilities from the FinOps Foundation Framework, you will add further measures of success that will need to be tracked and reported upon. The addition of further capabilities will also add extra functional activities that your staff will be expected to perform. Rely on your reporting to know when there is a task that needs to be completed and measure the outcomes that performing the task returns.

Further, as your adoption of cloud services increases, the amount of information needing to be presented will increase. Often teams focus on core cloud services, e.g., compute, database, and storage, but as your organizational cloud maturity increases,

expect to see a longer list of cloud native services used by your teams, such as Function as a Service (FaaS), machine learning (ML), or higher-level application services. Each of these added services will introduce different billing structures. Some bill by time like per second or per month; others charge based on quantity, or even have a little bit of both. The more services used in your cloud bill, the more complex the task of reporting on cloud costs becomes.

Multicloud

As your organization begins to significantly use multiple cloud providers, your FinOps team might start with separate reporting, especially if you choose to use each of the individual cloud service providers' billing tools. Although many of the terms and concepts used by the major cloud providers are functionally similar, there are differences in the naming conventions and service categorizations that can be confusing, and some elements are unique to a particular cloud provider. A good breakdown of differences between tagging and metadata structures in the cloud can be found in Chapter 12.

Separating data from different clouds forces your users to recall information between various environments and connect similar concepts themselves. As Edward Tufte said, "Comparisons must be enforced within the scope of the eyespan."

To reduce the effort and confusion associated with multicloud reporting, it is better if you combine the cloud data into consistent reports where the complexity of joining the information has been handled correctly. To achieve this, you will likely need to move beyond native tooling to in-house or external vendor tooling.

Data ingestion and normalization is a foundational capability of the FinOps Foundation Framework to enable other capabilities. Effective FinOps practice requires access to regular streams of detailed usage, utilization, and cost data, which can be categorized and analyzed to drive decision making. Monitoring platforms, security platforms, and business operations applications can also provide data that will inform on utilization, location, value, and usage, often at similar levels of volume and granularity. The data ingestion and normalization challenge grows as the complexity and diversity of the data increases. Be sure to look at the measures of success for this required capability presented with the Framework when determining how to handle your own data.

Putting Data in the Path of Each Persona

FinOps data is required to maximize the value of cloud spend, and unfortunately, it is often hidden away in isolated FinOps-specific reports and tools. When these reports

are orphaned in a dashboard that is outside of daily workflows, they are underutilized by the very engineering teams, business leaders, and finance teams who need them. To counteract this, aim to put *data in the path of* each persona's already established ways of working.

Data in the Path of Finance

The first persona to see some form of FinOps data is often the finance team. This comes from a mutual interest as finance seeks to have cloud spend data added into their established financial reporting systems. Cloud spend is first connected to company purchase orders to help with assigning spend to the right budgets. Getting FinOps data into finance systems allows your existing finance processes to work with your summarized cloud spend, and it reduces friction with finance teams who have well-established reporting and auditing requirements from which they typically cannot deviate.

Data in the Path of Leadership

Leadership and executive reports should also include FinOps data showing cloud spend data in relation to existing business metrics. For business leaders to make decisions, ensure key information on cloud spend, savings, and opportunities are present among other data they are used to digesting. Separating FinOps data from other business data reinforces the incorrect perception that FinOps is somehow disconnected from the rest of the business. As we'll cover in Chapter 26, you should aim to provide *unit economic metrics* in leadership reporting that marry cloud spend to business metrics to showcase the value and impact of cloud usage to the business, not just the cost.

Data in the Path of Engineers

Last, but arguably most important, we strongly recommend you work on getting FinOps data into the path of your engineers. When your engineers start thinking about FinOps, it should not be a drastic departure from their well-established daily rituals, i.e., you don't want them moving away from doing DevOps and spending time *doing* FinOps. Ideally, it becomes an integrated and required effort in one cohesive process.

We often say, *cost is just another efficiency metric*. Engineering teams will have dashboards and rituals focused on a long list of key metrics. These dashboards will monitor security, performance, and reliability metrics to manage their services running in cloud successfully. Aim to add key FinOps data into these existing locations and systems. Introduce the data they need to begin considering cloud spend changes—such as actuals versus forecast and opportunities to reduce costs—into their ongoing sprint planning. Isolated FinOps data and conversations create cognitive load and

distraction that invariably results in reactive cost work that occurs only when mandated. As a result, engineers will see FinOps tasks as interruptions to their work instead of being a part of it.

When introduced early on in the development lifecycle and into existing processes, FinOps data can encourage them to consider the cost at every step of designing a new architecture or making improvements to code. Engineers need to allocate the right amount of time for FinOps activities to achieve the business's desired outcomes around cloud efficiency.

Connecting FinOps to the Rest of the Business

> *Listening is more than being quiet while the other person speaks until you can say what you have to say.*
>
> —Krista Tippett, Journalist

Rather than reinventing the wheel, there are likely already teams that produce finance, executive, and engineering reporting with whom the FinOps team can partner to integrate their own data. This provides an opportunity to educate these personas about FinOps and continue the work of breaking down barriers between discipline silos within the organization. It also provides an important touchpoint to tackle the UI concepts discussed earlier in this chapter, ensuring that language is consistent and data is presented to encourage action. These existing groups are often wizards at using APIs to pull and massage complex data into readable and consistent reporting.

The idea behind the *Data in the Path* concept is not to take the whole world of FinOps and drop it into the middle of everyone's reports. Only bring the critical information that enables targeted conversations and decisions required for FinOps without interrupting current rituals. When questions arise from this information requiring more in-depth reporting and analysis, that is the time to connect your staff to the more detailed reports you have available in your FinOps arsenal.

Your job is to ensure each respective team has the context to truly understand cloud consumption data and, when faced with it, make the right assumptions leading to the right actions for the business. One can glean only so much from raw data regardless of how well it's integrated into other workflows.

Seek First to Understand

It may be tempting to look at an optimization recommendation or some form of efficiency scoring data and assume that the owner of that infrastructure isn't thinking about costs, is ignoring efficiency, or is simply a bad actor. Instead of making assumptions, use FinOps data to start a blameless conversation that seeks first to understand decisions before attempting to influence them.

Kim Wier, Director of Efficiency Engineering at Target, uses this model well. Her central FinOps enablement team goes to the various engineering teams and asks them to tell her what efficient usage of their resources should look like. "It's important to ask them how they want to see efficiency metrics," she says. "The engineers want to utilize their infrastructure as best they can in the best interest of Target." She can then provide the relevant FinOps data aligned to the business goals of that particular team. Kim goes on to say, "We aim to output recommendations based on what our platform partners view as important to the platform. If we make a recommendation and the engineering team has a strong reason why they can't do it, then we want that feedback loop to come back into our recommendations."

Stories from the Cloud—Jason Rhoades

The UI/UX of FinOps is not always read-only. Information in reports and dashboards often flows from the data source to the end user, but there can be information coming from those end users back into the system.

Jason Rhoades, Development Manager at Intuit, shares:

> For Intuit, a big evolutionary step was moving past the read-only concept of reports, with the introduction of FinOps related workflows that involved our end users performing write activities. These could be for something as simple as conducting an approval workflow to get some cloud waste excluded/hidden from a report. Or a contract between two business unit groups preceding a new cross-charge that affected how the report was structured. Or a team adjusting how they allocate their cloud assets together for the purposes of forecasting and reconciliation. As we began to go beyond simple reports to enable these sorts of capabilities, we found that we could solve a lot more problems, more comprehensively. But it did require growing the central FinOps team and adding some new skill sets to enable more interactive and iterative reporting capabilities.

This goes counter to the flawed approach that Noel Crowley from Fidelity described on the FinOpsPod as the most common one taken by immature practices: "What typically happens is that a FinOps team takes optimization recommendations from their native or platform tools to the teams and tries to get them to take action on them. Those on the receiving end of those recommendations then push back with a variety of reasons why the change doesn't make business sense, saying things like 'Yeah, I could rightsize, but I don't because XYZ.'" Conversely, Kim's approach at Target enables her engineering partners to lead her to the solution that makes the most sense for their portion of the business, rather than trying to overlay an ineffective standardized approach on everyone.

Conclusion

The reports and dashboards you create are some of the most important tools in building a collaborative FinOps practice. By thinking of your reports as the UI of FinOps, you can use common UI design considerations to make them more effective at communicating the right information to enable positive actions to be taken.

To summarize:

- Treat your reports and dashboards as production, with any changes going through testing in a staging environment before being deployed.
- Trust in the quality of your reports and dashboards is paramount for FinOps success.
- Consistency in reports avoids confusion or confirmation bias.
- Consider conditions that will impact how your reports will be viewed.
- Carefully select the information you present in reports to best influence the outcomes you want.
- Design your reports based on the perspectives of the intended audience. Reassess the design decisions you have previously made as your cloud journey progresses.
- To make FinOps part of the business culture, bring FinOps data into the path of existing reports and rituals used by engineers, finance, and leadership.

This completes Part I of this book. We've laid the groundwork by introducing FinOps, why you need it, how it requires a cultural shift, what insights the cloud bill provides, the FinOps Foundation Framework, and finally the UI of FinOps. Part II gets into the fun part: implementing the FinOps lifecycle within your organization.

PART II
Inform Phase

Now that you understand the fundamentals of FinOps, it is time for the fun stuff. In this part we cover the inform phase of the FinOps lifecycle, walking you through the finer details of allocating costs, forecasting, surfacing changes in your bill, and monitoring the performance of your optimizations. This is the phase of the lifecycle that drives accountability to your teams and readies you to set goals for future improvements.

CHAPTER 9

The FinOps Lifecycle

Back in Chapter 1, we discussed the core principles of FinOps. The principles are great to help guide actions, and in Chapter 7 we discussed the FinOps Framework, which was built on those principles. This chapter covers the ongoing, iterative phases of the FinOps lifecycle and how to apply them to the capabilities of FinOps. The principles haven't changed much since the first edition of this book. But we have seen a lot of different companies and organizations begin to build and iterate on top of them, making them their own. Much like the FinOps Framework, which is a set of building blocks meant to be assembled in various ways relevant to your organization, the principles are also open source bedrocks for you to assemble and build a FinOps plan for your organization.

The Six Principles of FinOps

Again, the principles are as follows:

- Teams need to collaborate.
- Decisions are driven by the business value of cloud.
- Everyone takes ownership of their cloud usage.
- FinOps reports should be accessible and timely.
- A centralized team drives FinOps.
- Take advantage of the variable cost model of the cloud.

Let's look at how each principle plays out in action with real-world ramifications, and later we'll dig into how specific framework capabilities are designed to leverage them to achieve specific results.

#1: Teams Need to Collaborate

First and foremost, FinOps is a cultural change that focuses on breaking down the silos between teams that historically haven't worked closely together. When done well, the finance team uses language and reporting that moves at the speed and granularity of cloud, product managers fine-tune their application scaling forecasts to accommodate expected income from new features, while engineering teams consider cost as a new efficiency metric.

At the same time, the FinOps team works to continuously improve agreed-upon metrics for efficiency. They help define governance and parameters for cloud usage that provide some control, but focus first on ensuring innovation and speed of delivery can flourish alongside cost efficiency.

The addition of a blameless culture to this principle enables the company to learn from mistakes. Taking away the need for a person/team to blame for a cost overrun allows a postmortem to focus instead on how an overrun can be avoided in the future and what changes the organization needs to make to learn from this incident.

> *If a culture of finger pointing and shaming for "doing" the wrong thing prevails, people will not bring issues to light for fear of punishment.*
>
> —Betsy Beyer et al., *Site Reliability Engineering* (O'Reilly)

#2: Decisions Are Driven by the Business Value of Cloud

Think first about the business value of cloud spend, not the cost. It's easy to think of cloud as a cost center, especially when the spend reaches material levels. The cloud is a value creator, but the more you use it, the more cost it will incur. The role of FinOps is to help maximize the value created by that spend. Instead of focusing on the cost per month, focus on the cost per business metric, and always make decisions with the business value in sight.

#3: Everyone Takes Ownership of Their Cloud Usage

Cloud costs are based on cloud use, which comes with a straightforward correlation: if you're using the cloud, you are incurring costs and thus are accountable for cloud spending. Embrace this fact by pushing cloud spend accountability to the edges of your organization, all the way to individual engineers and their teams. And give them the information and guidance to be able to do this important job for the organization.

#4: FinOps Reports Should Be Accessible and Timely

In the world of per-second—or even microsecond—compute resources, unlimited cloud storage, shared Kubernetes clusters, automated deployments, and services that can incur costs based on externally controlled triggers, monthly or quarterly reporting of cloud spending isn't good enough. Real-time decision making is about getting

data—such as spend changes or anomaly alerts—quickly to the people who deploy and manage cloud resources.

As discussed in Chapter 8, FinOps data should be put in the path of the people making infrastructure decisions, informing them of the information they need without adding the extra effort for them to find it. Real-time decisions enable these people to create a fast feedback loop through which they can continuously improve their spending patterns, make intelligent decisions, and improve efficiency.

Focus relentlessly on clean data to drive decisions. FinOps decisions should be based on fully loaded and properly allocated costs. The costs should be amortized to include any prepayments made as part of commitment programs and should reflect the actual discounted rates a company is paying for cloud resources. They should also equitably factor in shared costs and be mapped to the business's organizational structure. Without these adjustments to your spending data, your teams will make decisions based on incorrect data and hamstring value creation.

#5: A Centralized Team Drives FinOps

Cultural change works best with a flag bearer. A central FinOps function drives best practices into the organization through education, standardization, and evangelism. This centralized team is where you find the subject matter experts who make the changes to culture through advocacy and education. The FinOps team improves the available data via better tooling and modifies the business processes that enable your organization to *do FinOps*. You maximize the results from rate optimization efforts by centralizing them, which gives your teams on the edge the freedom to maximize the results from usage optimization. Remember, the most successful companies decentralize responsibility to use less, and centralize responsibility to pay less.

FinOps practitioners use performance benchmarking to provide context for how well their organization is performing. Cloud performance benchmarking gives a company objective evidence on how well it's doing. Benchmarking lets teams know whether they're spending the correct amount or whether they could be spending less, spending differently, or spending in a better way. Companies should use both internal benchmarks to determine how individual teams compare to each other in key areas such as optimization, and external benchmarks based on industry standards to compare the company as a whole to others like it.

#6: Take Advantage of the Variable Cost Model of the Cloud

In the decentralized world of the cloud, planning for capacity moves from a forward-looking "What are you going to need to cover demand?" perspective to a real-time "How can we stay within our budget given what we're already using?" perspective. Instead of basing capacity purchases on possible future demand, base your rightsizing, volume discounts, and RI/SP/CUD purchases on your actual usage data. Since

you can always purchase more capacity to fit demand, the emphasis becomes making the most out of the services and resources you're currently using, and to use as few of them for as long as possible.

As your cloud practice matures, aim to take advantage of cloud native services that scale with demand and models such as spot instances that can leverage low-cost resources when they are needed.

The FinOps Lifecycle

Now that we've detailed the core principles, let's explore how they're implemented across three distinct phases: inform, optimize, and operate (see Figure 9-1). These phases aren't linear—you should plan to cycle through them constantly:

1. The *inform* phase gives you the visibility for allocation and for creating shared accountability by showing teams what they're spending and why. This phase enables individuals who can now see the impact of their actions on the bill.
2. The *optimize* phase empowers your teams to identify and measure efficiency optimizations, like rightsizing, storage access frequency, or improving RI coverage. Goals are set upon the identified optimizations, which align with each team's area of focus.
3. The *operate* phase defines and implements processes that achieve the goals of technology, finance, and business. Automation can be deployed to enable these processes to be performed in a reliable and repeatable manner.

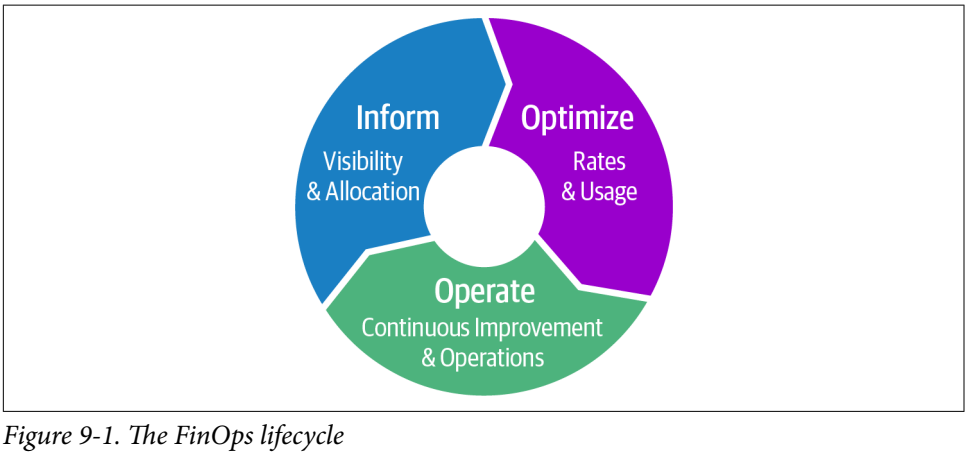


Figure 9-1. The FinOps lifecycle

The lifecycle is inherently a loop and is never complete. The most successful companies take an approach of gradual improvement and get a little better each time they go through it, building muscle memory through repetition and practice.

In each phase of the lifecycle you will take actions and perform activities based on the state of your cloud spend and your FinOps practice. Chapter 7 covered the FinOps Foundation Framework, which describes these aspects of the FinOps operating model. We described this using the metaphor of a garden where you will pivot to the tasks that need doing each day—watering, mulching, and weeding—based on the state in which you find the garden at that time.

The inform phase is where you will look at the state of your FinOps garden. The optimize phase is where you will look at which of the capabilities has the actions you might perform to make your FinOps garden healthier. And the operate phase is where you will take those actions in your organization's environment to ensure your garden flourishes.

Some of the capabilities lend themselves well to a certain phase of the lifecycle, and others might be used in more than one or all the phases.

Let's review each phase and some of the actions you'll take as you loop through the lifecycle. It's important to note that you won't perform all actions during every pass through the lifecycle. Chapter 1 covered the Prius Effect, which translates real-time feedback loops into data-driven decision making. Similarly, each pass through the lifecycle should be informed by the latest data you have to focus your efforts on the smallest set of most important activities needed at that time.

Inform

The inform phase is where you start to understand your costs and the drivers behind them. By giving teams visibility into their costs on a near-real-time basis, you drive better behavior. During the inform phase, you get visibility into cloud spend, drill down into granular cost allocation, and create shared accountability. Teams learn what they're spending on what services, and why, by using various benchmarks and reporting. For the first time, individuals can see the impact of their actions on the bill.

Some of the activities in this phase include:

Map spending data to the business
: Before you can implement accurate chargeback, you must properly map spend data to the organizational hierarchy by cost centers, applications, and business units. Tags and accounts set up by engineering teams are often not aligned to the view of the world that finance teams need, nor do they include the proper roll-ups that executives require.

Create showback (and other) reporting
: To push spend accountability to the edges of the organization, you must show each group what they are spending via a showback or similar model like chargeback. This visibility is the essential building block of driving ownership of spending and ultimately deriving the most value from it.

Define budgets and forecasts
: The FinOps team should provide the data needed for teams to generate forecasts of cloud usage for different projects and propose budgets for each. These budgets and forecasts should consider all aspects of a cloud architecture, including cloud native services, containers, and related costs. Managing teams to budgets lets you know when to lean in with optimization or spend remediation help. It also enables a conversation about why spending has changed.

  Forecasting of spend should be done for each team, service, or workload based on fully loaded costs and properly allocated spending, with the ability to model changes to the forecast based on different inputs such as history and cost basis.

Define your account strategy
: The way your organization distributes and uses AWS accounts, Google Cloud projects, Azure subscriptions/resource groups, or other hierarchical groupings will have a large impact on how you can do cost allocation, one of the critical capabilities used in the inform phase. A lot of the heavy lifting of cost allocation can be accomplished by having a consistent and logical way of distributing accounts so they can be used as boundaries for cost, as they are for security, resources, and other technical purposes.

Set tag strategy and compliance
: Your metadata strategy (tagging and labeling), covered in more detail later, is both an art and a science. Even with a strong account hierarchy, it's critical to get early alignment on a supporting tag strategy to get more granular. Without this, tag definition is left to the masses (or left out), and tag sprawl quickly makes any tags unusable.

Identify untagged (and untaggable) resources
: There are two types of organizations: those who have untagged resources and those who have fooled themselves into thinking they do not. Assigning untagged resources to teams or workloads—and applying a meta layer of allocation to untaggable ones—is critical to proper chargeback, visibility, and later optimization.

Allocate shared costs equitably
: Shared costs like support and shared services should be allocated at the appropriate ratio to responsible parties. There are a few methods of doing this, including sharing them equally or assigning them based on a usage metric like spend or compute hours. Leaving them in a central cost bucket is generally less desirable, as teams then don't see the true cost of their applications.

Dynamically calculate custom rates and amortizations
: Accurate spend visibility requires that companies factor in any custom negotiated rates, that discounts from RIs/SPs/CUDs are applied, and that amortized prepayments from RIs/SPs/CUDs are applied. This ensures teams are tracking the right spend numbers and aren't surprised if their finance team's bill doesn't match their daily spend reporting.

Analyze trending and variance
: Identifying spend drivers often requires ad hoc comparisons of time periods and the ability to report at a high level (e.g., cost center) all the way down to resources (or containers, functions, etc.) to understand cost drivers.

Create scorecards
: Scorecards let the FinOps team benchmark how different project teams are doing in terms of optimizing cost, speed, and quality. They're a quick way of looking for areas that can be improved, which should be done using the fully loaded and properly allocated cost mentioned previously.

Benchmark against industry peers
: Building on the concept of internal scorecards, more advanced FinOps teams extend their benchmarking to make comparisons with other industry peer-level spend data to identify their relative efficiency using a normalized set of spend characteristics.

Identify anomalies
: Anomaly detection isn't just about identifying expense thresholds—it's also important to identify unusual spikes in usage. Given the dramatic rise in the variety of variably charged services available from cloud providers, anomaly detection that watches for any deviations in spend helps you find the needle in the haystack that may need quick remediation.

Optimize

The optimize phase identifies measured improvements to your cloud and sets goals for the upcoming operate phase. Cost-avoidance and cost-optimization targets come into play during this phase, with cost avoidance being the first priority.

Processes are required to set and track the near-real-time business decisions that enable your organization to optimize its cloud. We'll also look at the cloud service provider's offerings that can help to reduce cloud costs. This phase includes the following activities:

Analyze KPIs and set goals
: It is during the optimize phase that you are looking at your KPIs and setting goals for how to achieve them incrementally. By understanding what you are trying

to achieve overall, you can set interim step goals to get there over time. This is helpful during the optimize phase because it helps to scope the opportunities you find and document for action in the operate phase later.

Find and report on underutilized services
: Once teams can see their properly allocated spend and usage, they can start to identify unused resources across all major drivers of spend (e.g., compute, database, storage, or networking). You can measure potential savings based on generated recommendations to eliminate things that aren't being used, to scale things that are used cyclically, to rightsize things that are chronically the wrong size, or to rearchitect things that are operating poorly.

Evaluate centralized commitment-based discount options
: As a cost-reduction measure, the FinOps team can evaluate metrics on existing AWS/Azure RIs, AWS SPs, or Google Cloud CUDs/Flexible CUDs to make sure the ones they have are being used effectively and then look for opportunities to buy more (or sell or modify existing ones). They track commitments and reservations, analyzing the entire portfolio across the enterprise to account for and understand the efficiency of usage and cost-avoidance actuals, complete with visibility into upcoming expirations.

Operate

Whereas the optimize phase sets the goals for improving, the operate phase sets up the processes for taking actions to achieve those goals. This phase also stresses continuous improvement of processes. Once automations are in place, management takes a step back to ensure spending levels are aligned with company goals. It's a good time to discuss particular projects with other FinOps team members to determine whether the team should make some changes. Here are some of the activities that take place during the operate phase:

Deliver spend data to stakeholders
: Creating the Prius Effect discussed in Chapter 1 requires stakeholders to regularly see how they're tracking against their budgets. Daily or weekly visibility gives them a feedback loop that enables them to make the right decisions for the business. During the operate phase you focus on how these reports are delivered to stakeholders, building out the processes and automation to generate the reports and make them available.

Make cultural changes to align with goals
: Teams are educated and empowered to understand, account for, and partner with other organizational teams to drive innovation. Finance teams are empowered to be bold change agents who move out of blocking investment and into partnering with the business/tech teams to encourage innovation. Each team must

constantly and iteratively improve their understanding of the cloud and level up their reporting efficiency.

Rightsize instances and services (or turn them off)
: During the optimize phase, you might discover that you're paying for more powerful compute resources than you need. Recommendations that have been generated are acted upon during the operate phase. Engineers review the recommendations and, where appropriate, make adjustments—for example, switching to less powerful, less expensive instances; replacing unused storage with smaller sizes; or using different tiers of storage that better match access needs. Mature FinOps teams do this across all major drivers of spend.

Define governance and controls for cloud usage
: Remember that the primary value of the cloud is speed of delivery, fueling greater innovation. At the same time, cost must be considered, so mature companies are constantly evaluating their agreed-upon guardrails on how and what types of cloud services can be consumed to ensure that they aren't hampering innovation and velocity. Overdo control, and you lose the core benefits of moving to the cloud.

Continuously improve efficiency and innovation
: These are ongoing, iterative processes of refining targets and goals to drive better business outcomes. We call it *metric-driven cost optimization* as discussed in Chapter 22. Instead of using a cadence for optimization actions (which are prone to inefficiency or human neglect), metric-driven cost optimization defines key metrics with target thresholds attached and monitored to drive future optimization actions.

Automate resource optimization
: Mature teams move toward programmatic detection of changes needed for incorrectly sized resources and offer the ability to automatically clean up underutilized ones.

Integrate recommendations into workflows
: Mature teams stop requiring application owners to log in to see recommendations and begin pushing items like architecture recommendations, rightsizing recommendations, modernization opportunities, and other opportunities to improve into sprint planning tools. Some of this information will be tactical cost-optimization data, but some may be strategic in nature, helping teams to understand how architecting or building their software can begin to build cost efficiency in ways that avoid costs.

Integrate chargeback into internal systems
: Once chargeback has been implemented and visibility given to teams, mature FinOps teams then integrate that data programmatically into their relevant

internal reporting systems and financial management tools via their application program interface (API).

Establish policy-driven tag cleanup and storage lifecycle policies
: Mature teams begin to programmatically clean up tags through policies like tag-or-terminate or tag-on-deploy. They also implement policy-driven storage lifecycles to ensure data is stored in the most cost-effective tier automatically.

Considerations

There are a few key considerations you should review in your FinOps practice when beginning your journey through the lifecycle. They fall along the key ideas of FinOps: having a clear understanding of your spend, creating a company-wide movement, driving innovation, and, ultimately, helping the business reach its goals.

You will want to evaluate the following:

Unit economics
: An important step is to tie cloud spend to actual business outcomes. If your business is growing and you're scaling in the cloud, it's not necessarily a bad thing that you're spending more money. This is especially true if you know what the cost is to service a customer and you're continuously driving it down. Tying spend metrics to business metrics is a key step in your FinOps journey.

  Unit economics provide a clear, common lexicon so that all levels of the organization can discuss cloud spending in a meaningful way. Instead of management setting arbitrary spend goals, it can set targets that are tied to outcomes. The management advice becomes "Don't worry about the total bill; just make sure you're driving down the cost per ride" instead of the more restrictive "Spend less on cloud."

Culture
: The operate phase is a good time to evaluate how well FinOps culture is being adopted. Problems such as inefficient utilization of resources or inadequate RI coverage are often due to poor communication and siloed organizations. Look for signs that teams are being proactive, designing cloud services that use the cloud efficiently. Assess if the available FinOps training is being completed by staff and that communication between finance and engineering is flowing smoothly.

Speed of delivery
: Speed of delivery is controlled by the trade-off between cost and quality. Management might want to discuss particular projects with FinOps team members to decide whether they want to adjust those two levers to see if they can increase the speed of delivery.

Value to the business
: Again, management may want to evaluate whether cloud spend reflects the value of the project to the business. This is another opportunity to discuss particular projects with FinOps team members to decide if they want to continue to operate them as they have been or if they can make some changes.

Where Do You Start?

You start by asking questions that kick-start the inform phase. Think of the FinOps lifecycle as a closed-loop racetrack—you can jump in at any point, and you'll eventually loop back around. However, we recommend you start at inform before you get into optimize or operate. Gain visibility into what's happening in your cloud environment and do the hard—but important—work of cleaning up your allocation so that you know who is truly responsible for what before you start making changes.

And no matter where you are in the lifecycle, you should be continually focused on culture and governance. The true power of FinOps comes from combining the actions and tools with cultural shifts that change how your whole organization relates to using the cloud.

Whatever you do, don't try to boil the ocean. A few years ago, we saw a major retailer try to go from 0% to 80% RI coverage in a single purchase. The company studied its infrastructure, consulted its engineering teams, checked its operating systems, and made a $2 million purchase. Managers high-fived each other on their awesomeness and then went back to work for the next few weeks. The next month the cloud bill was considerably higher, and the VP was furious. Upon review, the company found it had purchased the wrong RIs in the wrong operating system due to naiveté about how BYOL ("bring your own license") models are applied. That same retailer is now at 80% coverage, but it took a multiyear effort to uplevel finance teams and business units who were gun-shy after the earlier disaster. Take your time. Like anything, building competence takes time but can be accelerated by learning from those who have gone before you.

You Don't Have to Find All the Answers

Before you start telling teams to turn off this resource or downsize that one, you must get a true sense of what the cost drivers are and let the teams see the impact of their spending on the business.

This will drive some surprising, autonomous results. We learned about a great example of this via a Slack message from a team member, where a manufacturing company enabled six-figure-a-year savings simply by showing a team what they were spending (see Figure 9-2).

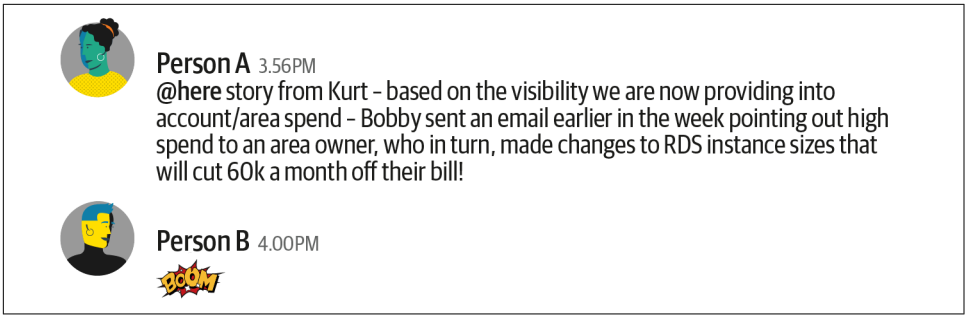


Figure 9-2. A real conversation about the results of cost visibility

The best part of this story is that FinOps didn't make any recommendations to the team. All they did was shine a light on the team's cloud usage. The team took charge to make improvements based on their understanding of the infrastructure cost. This is why you push reduction of usage out to the teams responsible for spending.

Conclusion

Remember, mastery of the FinOps lifecycle is an iterative approach requiring years of education and process improvement in a large enterprise.

To summarize:

- The FinOps lifecycle comprises three main phases that you continuously cycle through.
- Improve with each iteration of the lifecycle—don't try to do everything at once.
- Involve all your cross-functional teams early and often so they can learn with you.
- Constantly look for opportunities to refine your processes, but move quickly from phase to phase.
- The most critical thing you can do is provide your teams with granular, real-time visibility into their spending.
- Before you can do anything else, you need to fully load and allocate your costs, factoring in your custom rates, filling allocation gaps, distributing shared costs, remapping the spend to your organizational structure, and accounting for amortizations.

This may sound like a lot of work, but it's actually an easy process to get started. Next up, we'll work through the first phase of the lifecycle so you can start addressing the questions you need to answer.

CHAPTER 10

Inform Phase: Where Are You Right Now?

Begin your FinOps practice by asking questions. That's what the inform phase is all about. As you find answers to those questions, you can start evaluating the state of your cloud. In this chapter, we'll look at some of the questions you should start with, and we'll get an aspirational glimpse of what great looks like across some common capabilities. All of that will help you know where to focus when you get to the optimize phase.

As we've stated, FinOps isn't a linear process; it's a cycle. The visibility you get in the first phase is essential to equip you to enter the next phase. You'll spend the majority of your time in the inform phase. It's easy to make costly mistakes if you jump to optimize too quickly. As carpenters have reminded us since the craft of building began, "measure twice, cut once." Measuring informs you so you can make changes, and then you measure those changes against your operational metrics, and so on. You'll always circle back around to inform them to check in again on which actions will be beneficial to the business.

The first question you should try to answer is: "What's the total cloud spend, its forecast, and its growth rate?" If those numbers are immaterial to the business, it may be time to consider pausing after the most basic reporting is in place. Recall the informed ignoring concept covered in Chapter 2.

Data Is Meaningless Without Context

Of course, finding the data isn't enough. You have to interpret it. Your goal is to educate yourself. You need to start conversations with your colleagues in finance, engineering, and the line of business (LoB) about what you want to accomplish.

To do this, skilled FinOps practitioners will ask questions that will set them up to build the appropriate allocation constructs, as discussed in Chapter 11. Remember, you're not trying to boil the ocean. You want to identify questions that will be answered throughout the FinOps lifecycle.

This process also refines your common language that we discussed in Chapter 4. It prevents a lack of trust in the data, as well as the finger-pointing that often follows when there isn't alignment between teams.

> *Any improvements made anywhere besides the bottleneck are an illusion.*
> —Gene Kim, *The Phoenix Project*

Even the most experienced FinOps practitioners—those who have built multiple mature practices—follow a gradual maturing of FinOps process when setting up a practice. Your initial goal is to get some easy wins, harvest the low-hanging fruit, build muscle memory, and gain credibility. Think of DevOps, not Waterfall. Think of Gene Kim, not Winston W. Royce.

Seek First to Understand

As you start this phase, you may immediately find some waste. It's understandable and natural to want to run out and fix it. However, ask first, "Where is the waste coming from?" This will help you to identify the root cause and prevent it from happening next time. In other words, you should resist the temptation to jump straight to making changes in your cloud estate and instead focus first on answering the questions.

One of the most effective ways to determine the questions you need to ask is to interview the various stakeholders in the organization. When you understand what they're concerned about, you can use that knowledge to build the context needed for your reporting and allocation constructs.

Here is a set of questions to help you get started:

- What do you want to report on? Is it cost centers, applications, products, business units, or something else?
- Where is the bulk of spending coming from, i.e., from which set of services?
- Are you going to do chargeback? Are you going to do showback? There are benefits in each, but either way helps to drive accountability.
- If you're going to centralize rate management, is the common good a priority, or is it more important to recoup costs by chargeback?
- Is seeing spending trends enough, or do you need to accurately charge back to the penny? Trends are the goal in the beginning; later, granular allocation becomes important.

- How will you account for changes in cost centers? Early on, you may have a spreadsheet or a configuration management database (CMDB), but later you will have a meta layer of allocation data in your FinOps platform to dynamically remap against your organization's ever-changing structure.
- How will you account for people or applications shifting between teams, or recombining into different teams? After they switch, how will you get the information that they'll now care about, since their slice of the data will be different?
- How will you notify people that there have been changes to the allocation constructs?
- What are the tags you really need? Early on it may be only three, but later there may be dozens. However, if you do expand to dozens, it's likely you have too many tags, and a future loop through the lifecycle might see you optimizing the number of tags based on the updated goals set in the optimize phase.
- Will you do things like "lunch and learns" from your CCoE to regularly present the best practices and get people excited?

As you move around the FinOps loop, you answer more questions. Are you efficient? You'll need to go through the loop to find out. And after a few times around, you'll realize that you never actually arrive at the end. You just get better, and your questions get deeper:

- Which teams are driving the costs? Are they being efficient? Can you link their costs to unit metrics?
- Do you have budgets in place for each team? Are you managing teams to those budgets? Are you able to do activity-based costing?
- What's the state of your tagging strategy? Look at what tags are in place and what they are reporting. Look at coverage gaps. Consider how to allocate untaggable or shared costs.
- How will you keep your allocation constructs in sync and up to date? A spreadsheet? Cadence-based syncing with your CMDB? API integration between your FinOps platform and your CMDB?
- What is the commitment-based discount strategy? What commitment-based discount programs are there? How well are they being used? How could they be optimized? Which new ones should you make?
- And then...*rinse and repeat*, repeatedly.

You pick the low-hanging fruit in each phase and then move on to the next. Then you readjust your goals and go back around. But you should always start with cost visibility. Then in the next cycle, you set basic budgets, discussed in Chapter 13. And the cycle after that, you're managing them.

Each time you ask harder questions, always making sure that the other teams are brought along with you. In fact, their education is arguably more important than the FinOps practitioner's skills. The winning FinOps culture is a village that works together, not a roomful of individuals who are trying to speed through some sweeping changes.

Organizational Work During This Phase

Beyond the work of spend data analysis, there's much organizational and cultural work to be done to create a FinOps culture. During this phase of a healthy FinOps practice, you will also focus on:

- Getting executives aligned around goals and the changes to your working models that cloud brings
- Helping engineers understand their expanded role in affecting the business, particularly around cost as a new efficiency metric to consider
- Ensuring the right skills are present on your team (*FinOps.org* has more detail)
- Bridge-building with colleagues in engineering, finance, IT, architecture teams, procurement, product teams, security, ITAM groups, and the like, evangelizing FinOps work internally via events like FinOps day, to help share the concepts, impact, and early wins
- Aligning with engineering teams (and helping to take work off their plates) by managing as much of the rate optimization as possible while ensuring they have the data needed to manage usage optimization

Transparency and the Feedback Loop

In Chapter 1, we explored the Prius Effect and the impact of real-time visibility on your behavior. In an electric car, the flow-of-energy display enables you to see how the choice you're making in the moment—one that in the past may have been unconscious—is impacting the amount of energy you're using. Likewise, in the inform phase, you rely on constantly arriving data to drive real-time decision making and build accountability.

Recently, we were asked, "Is near-real-time data needed for everything in the cloud?" We considered this question but couldn't think of a single example of a report that should be looked at only once a month. Things move far too quickly in the cloud, which is due less to the computers themselves and more to the human behavior that's driving cloud innovation. During early FinOps maturity, reports are viewed daily or weekly. As FinOps matures they're viewed on an agreed-upon cadence (or based on anomalies), and in very advanced FinOps practices they're viewed after an alert

showing that a metric you've set has crossed the threshold. Chapter 22 digs into what that advanced stage scenario looks like.

Spending great amounts of time and/or money on making your data more real time than it needs to be for the processes you have inside your organization can be a source of frustration for new FinOps practitioners. Aim for frequently updated cloud spend data, but don't go beyond what benefits your organization. Again, don't try to boil the ocean at the risk of optimizing at the bottleneck.

A critical capability in this phase is the ability to detect anomalies in cloud spend. Anomalies are spendings that deviate from a typical value (average), a slope (trend) over time, or a cyclical repeating pattern (seasonality). They are the proverbial needle in the haystack that can be hard to detect among the complexity (and size) of most cloud bills. But they can also really add up.

Stories from the Cloud—FinOps Community

A remote engineering team at a multinational pharmaceutical company spun up three x1e.32xlarge instances in Sydney for testing of in-memory databases. At the time, an instance of this size cost just over $44 per hour. The three instances together cost over $3,000 per day, or around $98,000 per month. These seem like big numbers until you consider that the team's monthly cloud bill was over $3,500,000. So this change would have resulted in a paltry 2% increase in spend and wouldn't have been easily visible in high-level reporting.

Further obscuring this spend anomaly, the central team had just purchased RIs for another set of machines, a transaction that effectively canceled out the spend delta of the new X1e instances. However, because the FinOps team had machine learning–based anomaly detection, they found out about the use of the large instances the same day and were able to have an immediate conversation about whether or not so much horsepower was needed. Unsurprisingly, it turned out that it was not.

Granted, this is a story of a more mature stage company. A less mature FinOps practice typically starts with simple daily spend visibility that shows teams their respective spend. Even that amount of visibility still begins to influence their behavior accordingly.

Luckily, because identifying anomalous spending is such a critical issue, all cloud providers and many third-party tools and platforms provide you with the ability to spot anomalies more easily. To manage anomalies effectively, you must:

1. Have a solid cost allocation strategy—covered in the next chapter.
2. Frequently look at—or programmatically triage—the anomaly reporting.
3. Assign anomalies to the appropriate team responsible for the spend.
4. Follow a process for investigating and closing out anomaly tickets.

It is unfortunate how frequently anomalies cost organizations huge amounts of money that could have been greatly reduced if they simply paid attention to the anomaly alerts or knew who to talk to after the anomaly was detected. Managing anomalies starts in the inform phase. If you are not looking at your anomaly reporting, stop reading and go turn it on, then come back and finish this chapter. It's that important.

Benchmarking Team Performance

Using scorecards that benchmark team performance is the best way to find out how teams compare. Scorecards allow you to find the least-performing and highest-spending teams and also give you insight into how you benchmark against others in the industry.

Scorecards should also show you opportunities for improvement against core efficiency metrics. They should give executives a view of the entire business (or a portion of it) to see how everyone is doing and be able to drive down to an individual team level for actionable data. They are the CxO's best friend and best weapon to effect change. Scorecards should drive efforts by the teams and unify the experience between disparate teams who are working on similar efforts. And scorecards help teams compete against each other.

In a previous FinOps Foundation call, Intuit's Dieter Matzion, now at Roku, shared his approach to scorecarding. The key items were:

- EC2 efficiency via a rightsizing score
- Rate commitment program coverage and efficiency
- Elasticity measure to determine how well teams were taking advantage of the ability of cloud to bring resources up and down, based on workload

In addition to giving each team individual scorecards, tracking their efficiency across multiple metrics, Dieter also created an executive-level view that rolled up the scores on a team-by-team basis. This type of visibility shined a bright light on areas of opportunity. And it drove improvement, showing again that no one wants to be on the worst offender list.

A recording of Dieter's presentation, in which he drills into the specific metrics used to benchmark teams, is available on the FinOps Foundation website.

What Great Looks Like

As in all disciplines, there is no shortcut to becoming a high performer, so take the following measures of greatness with a grain of aspirational salt. Organizational FinOps muscle will develop only over months and years of practice. Of course, you can do things to help speed up the process, but ultimately you need to put in the time required for learning and maturing the cultural changes within an organization. In fact, you can do your business a disservice by trying to go straight to high performance. It inevitably results in mistakes, which bring about unexpected costs.

Table 10-1 shows various levels of proficiency across key metrics. Quantitative data was taken from the more than $9 billion of public cloud spend in Apptio Cloudability's dataset in 2019, while qualitative data was taken from a survey of hundreds of cloud consumers by the 451 Group.[1]

Table 10-1. Metrics of low, medium, and high performers in public cloud

	Low performers	Medium performers	High performers
Visibility and allocation of cloud spend	Reliant on vendor invoices and manual reconciliation	>1 day for partial visibility with limited retention of granular historical data	<1 hour or near-real-time visibility of all spend with all current and historical data retained
Showback or chargeback	Inability to provide teams an accurate accounting of cloud spend	Cloud spend is allocated to teams based on estimated usage of resources	Teams understand their portion of cloud spend based on actual consumption
Team budgets	Teams have no budgets	Teams have budgets	Teams budget and track spend against budgets
RI and CUD management	0%–20% of cloud services purchased via reservations	40%–50% of cloud services purchased via reservations	80%–100% of cloud services purchased via reservations
Find and remove underutilized services	Every few months	Weekly	Automated and policy driven
Unit economics	May not use	More technical, team-based cost per compute hour, etc.	More business-focused cost per customer, etc.

High performers are able to ask and answer complex questions about their spend and forecasts quickly, they do what-if analyses on scenarios on changing deployments, and they understand the impact of those changes on their unit economics. Due to

1 451 Research, *Cost Management in the Cloud Age: Enterprise Readiness Threatens Innovation* (New York: 451 Group, 2019), *https://oreil.ly/71tao*.

the granular visibility and allocation of cloud spend, they know where to look for opportunities to optimize their operational metrics. Or they can identify gaps in their information, or automation to create to help them mature in their inform phase.

This high level of capability enables businesses to be more competitive through speed of innovation, gives management and teams a better understanding of costs and COGS, and enables more insight into pricing of services.

Conclusion

The natural resting state of the FinOps lifecycle is the inform phase. Here, you understand the current state of your cloud financial management, which will help you identify opportunities to perform optimizations and operational improvement in the next phases.

To summarize:

- Build context around your financial data in order to answer questions.
- Use data to monitor spend and plan for optimizations and efficiency.
- FinOps culture helps personas across disciplines to work together to answer important questions about usage and cost.
- Following the FinOps maturity model is important. Each time you pass through the inform phase, you will be able to answer even more complex questions about your cloud as you mature.

We've identified the questions, but without the ability to divide cloud spend into more granular cost groups, answers about your overall spending are only so useful. You need a method of identifying which costs belong to which group to perform team-by-team reporting, showback, budgets, and forecasts. Then you can more effectively benchmark teams and compare them to each other. This is the subject of the next chapter, in which we introduce the concepts of cost allocation.

CHAPTER 11

Allocation: No Dollar Left Behind

This chapter is about the strategic decisions that you need to make before allocating your spend. While cost allocation is about assigning cloud costs to the correct business units within an organization, there are also many benefits gained by reporting those allocations to the whole business.

Why Allocation Matters

The advantages of leaving no dollar unallocated, unassigned, or untagged are vast. First off, you can tell who is spending what. You remove typical cloud cost discussions that end abruptly with "That resource doesn't belong to me." But most valuable of all, you can begin to individually trend each team's forecasted growth (versus the whole company's spend) and set budgets for teams so they know how they are doing against their targets, which Chapter 13 covers in more detail.

Once you've established that, you'll begin to know where anomalies are coming from, as well as which area of business and which workload drove them. Knowing what a resource is doing helps us begin to optimize intelligently by taking into account the characteristics of the environment and the scheduled plans of that team. Is it production or development? How much do you want to push the boundaries of their respective CPUs? And so on from there. Meanwhile, teams receive a realistic daily view of their spending that ties to what finance teams are going to want to charge back to them at the end of the month.

> *You should be heading toward unit economics to give responsibility for the budget to the teams. Central teams are still doing the allocation, but you want to push accountability out to the teams at the edge via chargeback and showback to make it their responsibility. The central team is defining how to use the cloud efficiently.*
>
> —Michele Alessandrini, Head of Cloud Adoption and Governance at YNAP

Allocation enables you to answer the business questions around cloud spend that were posed in Chapter 10. In fact, cost allocation is a vital link between business value and cloud spend. As people become aware of their actions and of the effects those actions have, that awareness drives a leaner culture. And it is chargeback and showback that create accountability.

We hope it's becoming obvious that the inform phase is a necessary prerequisite to the optimize phase. Even so, we've seen all too often a cloud novice who will dive headlong into trying to cost-optimize their cloud environment, all the while ignoring the importance and complexity of understanding cost allocation drivers in the cloud. A solid hierarchical account strategy and a decent tagging strategy are the key elements of any cost allocation strategy. But just establishing these isn't enough to solve the issue of allocation, because allocation can be a moving target. Consider that untagged or untaggable resources constantly appear, shared costs must be accounted for, and there's always the very common possibility that—right around the corner—a reorg or a change in how things need to be reported can break an allocation strategy, despite how carefully built it is.

People also add complexity. Different people will want things in different ways. Finance people want to know where the spend is going and how they can tie it back in a meaningful way to SKUs and products. Engineers care about variable consumption, individual service costs, and how their day-to-day actions impact the bill. An engineering director is going to care about multiple views rolled up into a global view. Specialists may need reporting showing the details of usage of different resource types or resources using particular types of licensing. Context is everything. So as you build your strategy, you'll be keeping all of the different needs and points of view in mind to create allocation constructs that suit all these different personas.

And of course, let's not forget about business metrics. Your strategy must let teams tie business metrics to their spend, thereby enabling unit economics and conversations about the value of the spend to the business.

Amortization: It's Accrual World

Recall the matching principle introduced in Chapter 4: it states that expenses should be recorded in the period in which the value was received, not necessarily during the period the cloud provider invoiced them.

The accounting principle to consider here is whether to report on a *cash* basis or an *accrual* basis. The former implies that you report spending during the period in which it was incurred, while the latter states that you report spending during the period in which the benefit was realized. This applies to billing constructs such as certain commitment-based discounts that have an up-front payment, where a portion of the initial expense will need to carry forward to when the commitments are used.

This "carrying forward," or amortization, is the equivalent of the hardware depreciation concept that many in the on-premises world are familiar with. But here it specifically applies to intangible assets. Without amortizing the up-front cost forward to when—and where—it was used, confusion and doubt are introduced around the real-time spend data. Operators are lulled into thinking they are spending less than they are, and there's often an unhappy surprise later on when they receive a larger invoice from accounting, which has applied amortization before charging back.

In Figure 11-1, the cash basis reporting graph at the bottom shows how spending can look lumpy without amortization applied. The accrual basis graph at the top of the figure shows how spending trends become clearer once costs are amortized.

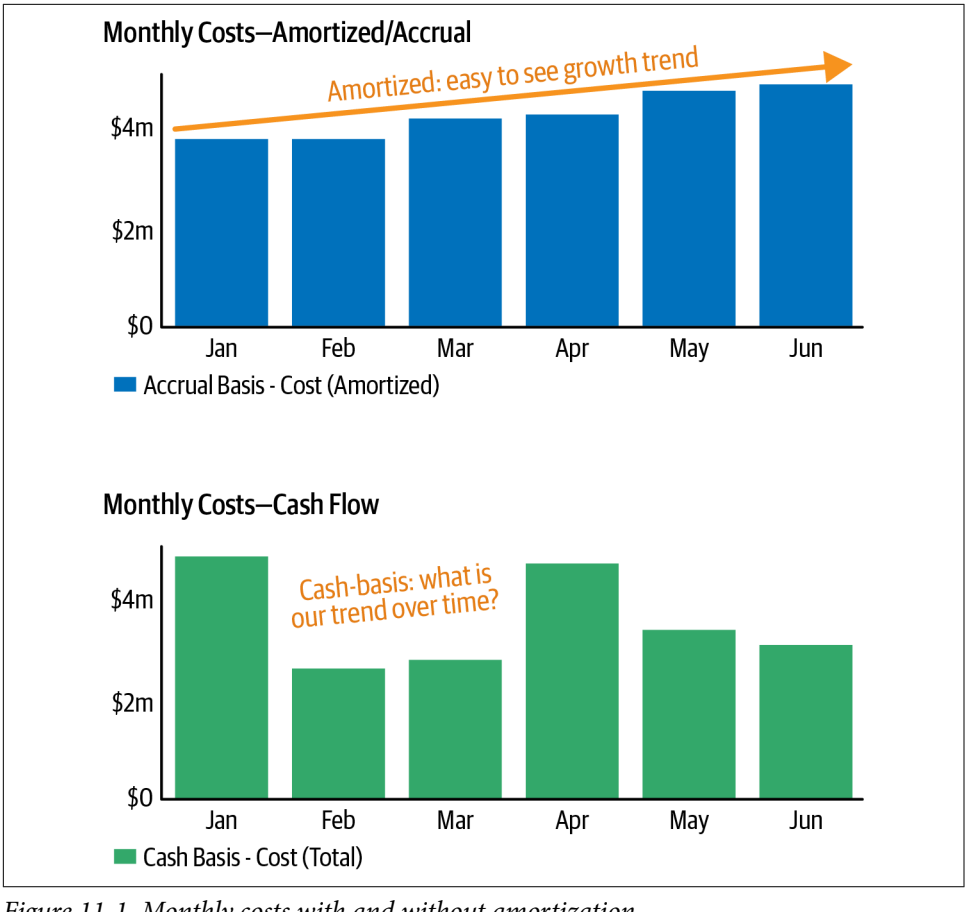


Figure 11-1. Monthly costs with and without amortization

Here's how the amortization and the resulting savings are calculated in a simplified example:

- On-demand rate for a resource is $0.10 per hour.
- A reservation costs $200 up front, and then $0.05 per hour when used.
- The amortized rate is $200, divided by 365 days of 24 hours each, plus the $0.05 per hour rate.

$$\frac{200}{(365 \times 24)} + \left(0.05\right) = \$0.073$$

- The savings works out to be the on-demand hourly rate minus the amortized rate.

$$0.10 - 0.073 = \$0.027 \text{ per hour (27\% savings)}$$

A company might have hundreds, if not thousands, of commitment-based discounts at any given point in time. This makes it quite challenging to track all active commitments. Each commitment would typically have a unique schedule, with a certain start and end date. In the case of convertible AWS reservations, the RI could be modified to a set of new RIs, requiring the original fixed price to be divided properly among all new target RIs.

There's more detail coming on commitment-based discounts of all sorts in Chapter 17.

It's recommended that amortization is factored into all spend reporting data, including forecasting, chargeback, anomalies, and basically any other data that's put in front of teams. If cost reporting and analytics do not match higher-level functions like forecasting and anomalies, data confusion can occur. And as we've stated a few times, the less data confusion, the better.

Creating Goodwill and Auditability with Accounting

The additional benefits of having a good account hierarchy and a good tagging strategy go beyond full cost allocation and accountability to auditability. Using auditability as an added bonus of tags can generate a lot of support from the finance teams as you push forward an allocation strategy.

> *A big part of accounting's job is making sure that all of the money spent can be audited and tracked. The more resources are tagged correctly, the more auditable is your cloud spend.*
>
> —Darek Gajewski, Principal Infrastructure Analyst at Ancestry

If that weren't enough, a solid tagging strategy allows you to build more accurate forecasting models, simply because you're using data that's richer, more accurate, and more granular. Since another key role of accounting is to forecast future expenses, tagging goes a long way toward building trust and goodwill between operations and finance.

All of this translates into less pushback, more willingness to move costs around, and, in general, teams that are better equipped to achieve a company's goals.

The "Spend Panic" Tipping Point

As is all too often the case, different teams running in cloud have little oversight until the spend gets to be high enough that it draws the attention of leadership. While there is often pushback against chargeback—and, to a certain degree initially, to showback—operating without showback/chargeback does not enable answers to the questions around who is spending what.

Figure 11-2 shows a sample journey into the cloud where cost management is not on the radar until spend draws executive attention.

The typical cloud maturity curve shows the point at which organizations start to pay attention to their cloud spend. The illustrative graph culminates in a very specific point: that moment when spending crosses an invisible threshold and the executive team starts to really care about it. For some organizations, that may be $1 million per month, while for others it may be $10 million per month. The number is different for each organization, but the output is the same. The tech organization is suddenly whiplashed into caring about—and seeking to optimize—cloud spend.

Spend panic is often such a surprise that there is a halt or pause in cloud use. This can hurt the company overall in terms of making progress on business objectives, as well as throw the cloud teams into a panic.

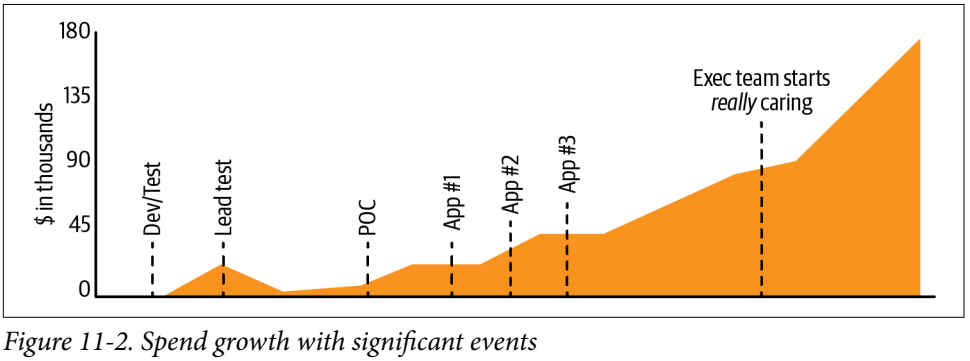


Figure 11-2. Spend growth with significant events

During a FinOps Foundation call, Ally Anderson, a business operations manager and eight-year FinOps veteran, tells a similar story about her own organization. There have been three different times when there was a "spend panic" that resulted in a push for more accountability:

> The first one drove a need for questions being answered. This got finance onboard with the idea of showing teams what they were spending. The spending increased again, and drove a period of shameback. A light was shone on teams who were increasing their cloud costs. This brought in showback for a spell, until another threshold was crossed that resulted in another period of shameback in an attempt to try to get teams to reduce spend. It wasn't until a third period of spending panic that the light bulb went off: the bill going up isn't necessarily a bad thing if our business is growing.

At that point, Ally and her organization began to reframe the question as: "Are we spending as efficiently as possible?" As she explains, "The conversation moved to 'how do we optimize?' instead of mandating a cost reduction target."

These conversations were possible because of the allocation strategy that had evolved through many years of maturing FinOps.

Spend panic is going to happen. If you have an automated system in place to see your spend patterns, and an approximate date for which you predict spend will increase by too much, you are prepared for the inevitable executive scrutiny that will follow. Without defensible cost allocations, you are heading into an avoidable conflict with business leaders. You can protect yourself, and your sanity, with a scalable, easily updated allocation strategy.

Spreading Out Shared Costs

Every organization has shared IT costs, and these span multiple business departments. As organizations increase their adoption of public cloud resources, it becomes increasingly difficult to assign shared cloud resources to specific business owners, to understand how to properly and fairly allocate costs, and to actually forecast future business unit budgets.

Support charges are a common example of this challenge. Cloud vendor support charges are generally applied at the parent account level. Some organizations choose to cover this charge with a Central IT/Cloud team's budget, but that approach isn't commonly used. More commonly, a Central IT/Cloud team is considered a supporting organization (being a cost center) and therefore needs to allocate its cost to its customers: business units or application owners.

Modern architecture has also introduced more shared costs with the rise of shared platforms. These platforms have multiple teams working with the same core resources, such as data lakes built in commonly shared S3 buckets or Kubernetes

systems running on shared clusters. At first glance, chargeback and showback for these platforms can seem impossible, but the trick is proper tagging and splitting shared costs correctly.

There are typically three ways to split up shared costs like this:

Proportional
: Based on relative percentage of direct costs

Even split
: Split total amount evenly across targets

Fixed
: User-defined coefficient (the sum of coefficients needs to be 100%)

Mature organizations tend to proportionally distribute shared costs among business units based on their direct charges. Let's consider an example. Table 11-1 shows how much each business unit of an organization consumed in a given month, and how the organization was charged $10,000 of enterprise support charge.

Table 11-1. Cost allocation example with centralized support charge

Business units	Cost	% of total (excluding enterprise support)
Sales operations	$50K	50%
Engineering—QA	$30K	30%
Engineering	$20K	20%
Enterprise support charge	$10K	
Total	**$110K**	**100%**

Without sharing the enterprise support charge, you can see that:

- Sales operations will be accountable for a total of $50,000.
- Engineering—QA will be accountable for a total of $30,000.
- Engineering will be accountable for a total of $20,000.
- The $10,000 enterprise support charge isn't being distributed among teams and would need to be assigned to a central budget.

In Table 11-2, the proportional sharing mode, the enterprise support charge ($10,000) will be distributed among the business units based on the percentage of their raw spend.

Table 11-2. Cost allocation example with amortized support charge

Business units	Allocation amount	Allocation percentage
Sales operations	$55K	50%
Engineering—QA	$33K	30%
Engineering	$22K	20%
Total	**$110K**	**100%**

Sharing the enterprise support charge, you can see that:

- Sales operations will be accountable for a total of $55,000 ($50,000 direct cost + $5,000 support charge allocation).
- Engineering—QA will be accountable for a total of $33,000 ($30,000 direct cost + $3,000 support charge allocation).
- Engineering will be accountable for a total of $22,000 ($20,000 direct cost + $2,000 support charge allocation).
- The enterprise support charge has been distributed and does not impact a central budget.

For more information on managing shared costs, the open source playbook on the FinOps Foundation website expands on this topic.

Chargeback Versus Showback

There are ongoing conversations in the FinOps community about when to do chargeback or showback.

Both approaches drive accountability, but you need to look at the core difference between them. Chargeback happens when the actual cost being incurred is allocated back to the budget or profit and loss (P&L) of an individual team. Showback happens when you show teams what they are spending, but the dollars are allocated internally from a central budget.

The best way we've seen to think about the difference comes from an article by Barry Whittle:

> *Showback leaves a bill on the table for analysis; chargeback leaves the same bill, but requires payment.*
>
> —Barry Whittle, Apptio

In practice, the choice between the two is more often informed by the accounting requirements of the organization rather than by its maturity. Enterprises will often choose one or the other due to an organizational bent toward or away from

departmental P&L statements. For instance, a finance leader or CFO may be against the concept of P&L across the various parts of the enterprise, or it may not make sense for certain parts of the business. In these cases, organizations will lean toward showback and not do full chargeback. There are often requirements to have separate accounting of costs for different business units, or different parts of the enterprise operating under different taxing authorities. These types of situations may require chargeback to occur.

But chargeback at too low a level can impose other difficulties too. It is common in the cloud to have adjustments, or credits, that appear after the close of the accounting period. Indeed, all usage is reported in arrears, so the need to regularly estimate end of period costs and make later adjustments can create difficulties for some finance and accounting groups. Angry accountants do not make good FinOps partners.

Stories from the Cloud—Rob Martin

The transition from showback to chargeback takes some time. Rob Martin, Director of Learning at the FinOps Foundation (he was formerly on the AWS Cloud Economics team) shared this story:

> If there's not a burning need to do chargeback for legal or tax reasons, the transition can take years to accomplish. At a large company I supported, it took three fiscal years to get to true chargeback. Their first year of cloud use was very exploratory: they spent most of it learning about how cloud worked, experimenting with their first applications, and building an understanding of just how different cloud budgeting was from their traditional model. Very little was done in budgeting or forecasting during year one because the organizational IT budget was already set, and the vast majority of their IT was still on premises. In their second year, the teams using cloud got a bit more flexibility but were still held to the traditional budgets. The FinOps team provided showback to each application team, but they were being funded from large traditional budget line items. In year three, the organization was able to finally clearly define how the cloud teams would control and flexibly manage their budgets throughout the year, and the finance team was confident that the P&L owners over those teams could be charged directly for their costs in the accounting system. The teams forecasting spend were confident in these forecasts because they'd had access to showback reporting over this entire time and had built the processes that would make them successful when the costs actually hit their accounting line items.

A Combination of Models Fit for Purpose

Ally Anderson shared that her company uses a combination of chargeback and showback. When it's a customer-facing SaaS workload, the company does a showback spend while centrally managing the budget. Then, for certain R&D workloads, they do a chargeback to the specific team's P&L.

Ally confirmed that allocating spend among teams is complex and ongoing. She maps each team's dozens of disparate accounts against the relevant products and cost centers that the business needs to report against. However, after a big reorg, she told us that teams rarely tear down cloud resources and rebuild them in different accounts to match the new organizational structure. But it's an important step in a healthy FinOps practice. You must take existing usage of cloud resources and reattribute them to the appropriate team, portfolio, VP, or cost center.

It's a common challenge that many FinOps teams encounter. The way engineers tag resources doesn't necessarily align to the way the business needs to report cloud spend. And spending often needs to be remapped to match the cost centers or products that are the rightful *owners* of cloud spend.

Accounts, Tagging, Account Organization Hierarchies

Each of the major cloud service providers have a range of elements that, when used together, enable you to build a successful cloud cost allocation strategy:

Account hierarchies
: How your engineers place their infrastructure into AWS accounts, Google Cloud projects, and Azure subscriptions forms the first layer of your cost allocation strategy. Individual accounts can be used to isolate costs of one environment or application from another.

Account naming
: Often overlooked in the early stages of an organization's cloud journey is having an account naming standard to record details about the account's use. Aim to include details like the product and environment that will be operating inside the account. An example name might be "*productA-production-01*." The FinOps practitioner should encourage the organization to have a central checkpoint to ensure this naming process is followed. All requests for accounts should go via a central process. In AWS, this would take the form of linked accounts or cost categories. In Google Cloud, this would be in the form of projects or folders. In Azure, it might be subscriptions or resource groups.

Tagging
: Make heavy use of tagging and labeling, especially in accounts that have multiple applications or business units operating inside them. A tag key that tracks the department or cost center for the cloud resources is especially helpful for FinOps. Tags help carve out spending inside accounts into different groups. Require that tags be applied to all resources, using various levels of proactive enforcement. However, you should know that not all spending is taggable and not all end users are compliant. So there is often still a sizable chunk of untagged spend that is reported out to management as a data hygiene metric.

Shared costs across accounts

A common problem that FinOps practitioners face is how to allocate central costs in the cloud world. You will need to split and spread out costs like volume discounts, enterprise support, and RI prepayments. Inside accounts shared costs for items including Kubernetes infrastructure, networking costs, monitoring, and logging systems will also need to be handled by FinOps. For more on the ways to split out shared costs, see the open source playbook on *FinOps.org*.

Returning to the story shared by Ally, in the early days, when they were using the multitenant account approach, tagging coverage was not as good and left most of their spend untagged. So Ally focused on reducing the ratio of spend to untagged spend over time. On an account-by-account basis, team leaders in charge of particular products have been given a granular view into the services that comprise those products. Near-real-time reporting is now provided, resulting in increased visibility into spend over the month and quarter. This has helped to close allocation gaps by shining a light on them.

Following a typical maturity curve in FinOps, Ally began by keeping cloud volume discounts, enterprise support charges, and prepayments like RI amortization out of the reporting that end teams received. Over the years, Ally's FinOps practice has matured, and her teams have factored more and more of those costs into their spend reporting. Today, her teams see the properly adjusted prices. They are able to look at their fully adjusted spending, with amortizations and accurate volume rates baked into the numbers, which gives them a more accurate picture of actual cost.

The Showback Model in Action

Essential to showback is clearly tying costs back so teams know how much they've spent, showing each team their actual spend against forecasted spend, including the variance between the two and a comparison to what was budgeted. Provide showback reporting on a weekly, monthly, and quarterly basis, as well as providing regular, more granular reporting into cost drivers and anomalies.

Aim to have your CTO run a monthly operating review with each of the engineering VPs to show their spend. When there aren't any glaring variances, these meetings tend to be quite brief. Larger variances bring a more focused approach. The FinOps processes should provide VPs an as-early-as-possible heads-up that they will overshoot their budget.

When a variance comes to light, quickly highlight the issues and the cost drivers that caused the variance, and then the VP can work with the team(s) to implement a fix without waiting for the monthly operating review. This is a great example of the breaking down of silos and of real-time decision making.

Chargeback and Showback Considerations

There are multiple ways to implement chargeback/showback, like passing costs straight through, recovering additional revenue, and covering administrative costs:

Charge back actual costs
: This is the most common method and is recommended for a cloud-first organization. There is no second set of books to maintain, and everyone is aligned around the same set of numbers and goals. Chargeback is done using full amortization of prepayment, with custom or discounted rates applied, and—in the case of AWS—using unblended rates to show the actual cost a team incurred in the period a resource was used. The most mature organizations also split up shared costs such as support charges.

Centralize commitment-based discounts and keep some portion of the savings
: This approach allows a central team to fund itself and also shares the benefits of savings with the teams. However, there may be some pushback from teams who have been required to give up control of RI/SP/CUD buying and thus are unable to access all the savings possible.

Centralize negotiated discounts and keep some portion of the savings
: Custom negotiated discounts with a cloud provider typically increase the more you commit to spending. The economy of scale a central team can get enables the best possible rates, at a level that individual business units cannot achieve. Sometimes these rates are highly confidential as well, and there's a desire not to share them broadly in the organization. For all these reasons, some companies choose to keep a portion (or all) of the negotiated rate savings to fund the central team's administration. While not uncommon, this approach goes against the transparency valued by mature FinOps practitioners.

Decisions about which showback or chargeback model to use are best made with a broad discussion of how your organization wants to show costs internally to different teams, how strict it needs to be in charging back direct costs, how entrepreneurial you want the FinOps team to be, what accounting rules are in place, etc. It is a conversation and discussion that must include finance and accounting teams and must ultimately be made at the executive level due to the impact of the decisions on the company's P&L.

Conclusion

Allocation strategy varies from company to company, but it is an important building block in the FinOps lifecycle. It enables you to focus on the right areas to ultimately answer the questions posed in Chapter 10 that ask what you are spending on cloud and who is spending it.

To summarize:

- Cloud spend in a FinOp practice is typically viewed on an accrual basis with amortized prepayments, discounted rates, and shared costs factored into the data that teams see.
- Aim to factor in shared costs to the model, such as support, amortization, and unallocated expenses.
- Chargeback and showback each drive accountability.
- A showback or chargeback model can also help improve tagging and auditability.
- Chargeback is the more advanced model, but it's not right for all companies due to accounting practices. It's often best to start with showback and then explore organizational support for moving to chargeback.
- Having a model in place before you hit a "spend panic" tipping point will help answer questions for executives about what's driving spend.

Now that we've covered the *why* of allocation, let's look at the *how*. In the next chapter, we'll dive into specific tactics and strategies for implementing tagging and account structures in the enterprise.

CHAPTER 12
Tags, Labels, and Accounts, Oh My!

In the previous chapter, we discussed why cost allocation is essential for FinOps. Whether it's done via tags, accounts, folders, or labels (or more likely a combination of all of these), cost allocation is the building block for all other FinOps capabilities.

Without a consistent allocation strategy, you end up with large chunks of unallocated costs with no way to split them up or to identify which team is responsible. How can everyone take accountability for their cloud spend (principle 3) if they can't easily see what it is?

The principal mechanisms for allocating cloud cost are dividing usage into:

- AWS accounts, Google Cloud projects, Azure subscriptions or resource groups, or one of the cloud-specific meta-groupings of constructs such as AWS Organizations, Azure management groups, or Google Cloud folders. These provide the cleanest and most easily enforced cost allocation definition, but lack granularity and flexibility.
- Resource-level metadata-like tags (AWS, Azure, or Google Cloud) or labels (Google Cloud only). These provide granular resource-level key/value pairs that provide deeper context for usage and billing data, but require efforts at runtime or later cleanup.

Throughout this chapter, we'll cover the core mechanisms for allocating costs and how they each contribute to cost allocation, and we'll describe successful strategies we have seen implemented at scale.

Cloud providers use different terms for similar concepts: tags, labels, accounts, subscriptions, projects, etc. This can be generically referred to as *metadata*, but for simplicity's sake, we'll refer to hierarchy structures as *accounts* and key/value pairs as *tags* in this chapter. So when we say *tags*, we're also referring to Google Cloud *labels*. When we say *accounts*, we're also referring to Google Cloud *projects and folders* and Azure *subscriptions* and *resource groups*.

Tag- and Hierarchy-Based Approaches

Depending upon the size of an environment, detailed cloud cost and usage data might reach millions or even billions of lines/rows. For every resource you have consumed during the month, there will be details about where the resource was run, how much of it you used, and the rate you were charged for the usage.

What's missing from all this detail initially is the resource's business logic: who owns it, who should pay for it, what business service the resource belongs to, and many other unanswered questions. Of course, cloud service providers cannot be expected to add this detail to a bill; all of these details are customer-specific. So, you use tags. Tags allow you to assign business-context details to specific resources and then to analyze that information to answer your company-specific questions.

Personas in every discipline in a company—finance, operations, engineering, and management—can gain insights from cloud usage data. Finance might use the data to apply chargebacks/showbacks to the correct business unit. Operations might break down cloud spending for budget tracking and capacity planning. Engineering may want to separate development costs from production costs. And management can use the data to understand which business activities are driving costs. Everyone uses it to filter and summarize the massive pile of usage data down to the manageable set of data they're looking to take action against.

The key to winning at FinOps is to get account and tagging strategies established early on and maintain them consistently. To do so, you need cross-team visibility. Fortunately, a variety of tools are available, depending on the cloud provider you're using. All cloud vendors offer a form of key/value pair tags that can be applied to a resource, as well as hierarchy-based account groups in which resources are run.

Some cloud service providers even provide solutions that allow you to group and build deeper hierarchy-based allocations. Google Cloud Folders, AWS Organizations, and Azure Management Groups can be used in addition, or even as an alternative, to tags on individual resources. AWS provides the ability to tag accounts, and Google Cloud allows tags on projects. Hierarchy-based allocations allow for grouping accounts into clusters that can themselves be tagged. This capability is constantly being improved by all the cloud providers.

The most successful FinOps practitioners we've seen tend to use a combination of both approaches—tag-based and hierarchy-based—in their allocation strategy, but doing so is not required. In fact, depending on the complexity of your infrastructure, you could possibly do allocation without tags/labels. Or you could do it without separating your usage into accounts. But as your environment becomes complex, you'll likely use both.

Apply them with consistency at any given time, though it is likely you will change your strategy over time as you evolve in both your cloud use and your FinOps maturity. It's OK to have several *models* of account or tag strategy working at once, and it's unlikely you'll ever get your environment to be 100% consistent. This is OK!

Generally, if you're going to start with only one strategy, a hierarchy-based (accounts/subscriptions/folders) approach works best for mapping to one (or a few) dimension(s), such as the company's business unit/department/team structure. Account structures are mandatory (all resources are necessarily contained within one and only one of them), they are less granular (fewer things to identify and allocate), they are more individually significant (users can spot their account names more easily than their resource names), and they're likely already used for isolation of resources for technical reasons like security.

Augmenting with a tagging strategy can be done over time, but starting with tagging only can be problematic. Time and time again, we've seen a tag-only approach lead to missing coverage, resulting in a lot of unallocated spending even after herculean efforts to increase tagging coverage. We found that not only are development teams notoriously bad at ensuring all of their resources are tagged (especially in early days of cloud use), but also the cloud service providers generally have a small percentage of resource types that do not support tagging at all.

Without everyone being consistent with the strategy, costs will be incorrectly allocated. For example, if you're using accounts as your cost allocation strategy, and you have teams that are sharing accounts, costs can end up being assigned to one group and not the other. The most flexibility and success comes from having an approach that encompasses both accounts and tags. By combining both a hierarchy-based and a tag-based strategy, you can use tags for the fine-grained details while using the hierarchy-based strategy to catch the remaining unallocated costs.

There are three core methods for adding cost allocation data to usage and billing data:

Resource-level tag
: Applied by engineers or the cloud platform provider directly to the cloud resources.

Accounts/projects/subscriptions
: Provided by the vendor and represented in the bill.

Post-bill data constructs
: Uncovered by manipulating the billing data and supplementing it with extra detail using third-party data or analytics tools or self-managed data processing.

Getting Started with Your Strategy

To successfully build a tag- and/or hierarchy-based allocation strategy within your organization, you must do three key things.

Communicate Your Plan

Before you start planning your tag and account strategy, you need to make sure everyone is involved. Without all the key stakeholders, there's a risk that you'll create a strategy that doesn't meet everyone's needs. Your goal is to create a strategy that's best for the whole company, not just for one or two teams. In fact, if teams have already implemented their own tagging or account layout strategies, you must start by auditing them to find out what is working and what is not.

Often, when there are existing engineering lead strategies, they don't have a financial focus. Your strategy must be financially inclusive, while taking into account the needs for cost allocation and future cost optimization.

For the critical financial splits, the best pattern is to have team, service, business unit/cost center, and organization. These divisions are used to group costs and resources to answer questions at each layer: team versus team, service versus service, and so on. Consistency is mandatory. The less important or more team-specific ones can be recommended but not enforced.

Keep It Simple

A cost allocation strategy can seem overwhelming, especially with complex infrastructure, so it's important to keep the initial strategy simple. We recommend starting with three to five obvious areas whose costs you want to understand. When you're leveraging tagging for cost allocation, you might initially focus on tags for the business unit, the product, the owner, and the role. Then you might ensure your development services are deployed into a different account than your production services. You use accounts to isolate your costs by environment. Even these small first steps yield big returns in terms of information. In fact, we've seen many times that an initial focus on top-level cost allocation can lead to major visibility and optimization wins.

Keeping things simple is important as you move forward, too. Your goal should be to make your system as intuitive as possible. Sometimes when companies get more granular and complex as a system grows, their FinOps practices end up with a few

large accounts that have a lot of teams operating inside. Trying to decipher later which costs are for which team/service/environment can be tough.

Formulate Your Questions

Since the whole point behind an allocation strategy is to answer key questions about how your company uses the cloud, it's important to define your questions—and to do it early in the process. Some common questions FinOps should enable you to successfully answer are:

- Which business unit within the organization should this cost be charged to?
- Which cost centers are driving your costs up (or down)?
- How much does it cost to operate a product that a certain team is responsible for?
- Are you able to establish which costs are nonproduction and safe to turn off?

Using these questions as a starting point, you can determine what elements are missing from your billing files (see Chapter 5) and then use that information to help guide your account hierarchy and tagging strategies.

Comparing the Allocation Options of the Big Three

Let's compare the three big cloud service providers to understand how your allocation strategy can be affected by their various offerings (see Table 12-1).

Table 12-1. Comparison of cloud vendor allocation options

	AWS	Google Cloud Platform	Microsoft Azure
Hierarchy	Accounts (management and member)	Projects	Subscriptions and resource groups
Meta hierarchy	AWS Organizations	Folders	Management groups, departments
Key/value pairs	Tags	Labels and tags	Tags
Number of tags per resource	50 (some services allow only 10)	64	50 (some services allow only 15)
Tags automatically allocated to your detailed billing data	No—manual selection required	Yes—some limits apply	Yes—some limits apply
Tag restrictions	Some services limit supported characters	Lowercase letters, numeric characters, underscores, and dashes	Some characters not supported

	AWS	Google Cloud Platform	Microsoft Azure
Tags can be applied to	Most services' resources (constantly changes; see AWS documentation for all current details), accounts (via AWS Organizations)	Most services' resources, projects, and folders	Most services' resources, Azure resource groups

From Table 12-1, you can see that the granular allocation strategy is based on how each cloud service provider allows you to isolate resources across its platform. Each cloud service provider has a way of tagging (although Google Cloud offers both tags for policy definition and labels for allocation), but there are differences in the limits each provider applies to these tags.

Some providers support tagging at the account or resource group level. All Azure resources are created within resource groups, and tags can be associated with the resource group as well as with the individual resources within the group. Similarly, AWS Organizations allows tags to be associated with specific accounts. There are various ways to use and view these layers of tags within tooling from the cloud service provider and in your own FinOps tools.

Later in this chapter, we'll cover the considerations you need to take into account around these tag restrictions, especially when you're currently using multiple cloud service providers (or are likely to in the future). It's important to realize that for some cloud service providers' tags there are actions required to ensure that the tag values are reflected in billing data.

Comparing Accounts and Folders Versus Tags and Labels

Tags and accounts each have advantages and disadvantages. As we've mentioned, it's generally better to use them together than to choose one or the other. Tags are highly flexible, but 100% coverage can be difficult, if not impossible, to achieve. And though allocation based on account separation can give you clean chargebacks, it has limited reporting options.

Accounts are mutually exclusive, whereas tags are not. Accounts should be your primary index, and labels are secondary. A common best practice structure is to use accounts to split out the most important primary divisions and then use tags to dig down deeper into those accounts for more fine-grained visibility.

In Figure 12-1, we're using accounts to break out individual products and their respective environments. This is important, because different products or applications often represent different cost centers or budgets. Separate environments (development, production, staging, etc.) may need to be accounted for differently by the finance team, as some are COGS expenses and some are R&D. Further, various teams may want to group together production versus development spending to see where money is going or to apply different policies for cost optimization.

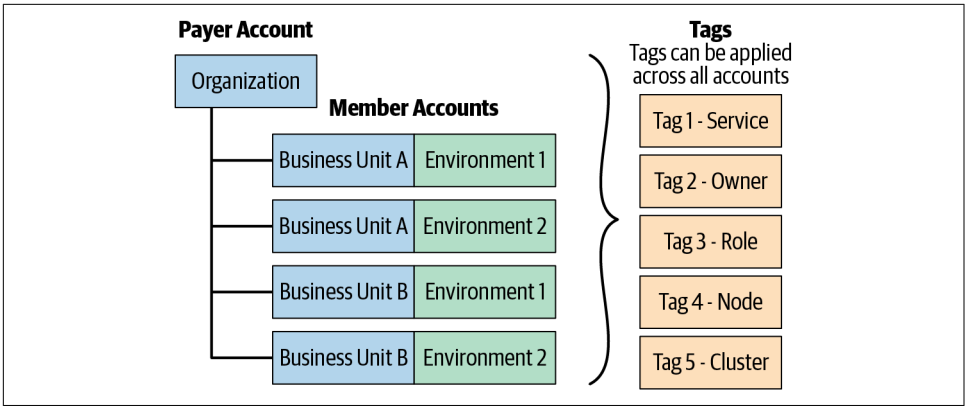


Figure 12-1. Common cost allocation strategy for AWS accounts

In some companies, tags are used as the primary allocation approach for a limited number of accounts. This tends to be more common in companies with fewer products and less stringent requirements for breaking out cost centers. The challenges tag-first companies often face comes when they apply a broad tag policy to cover as many resources as possible. At reporting time, they're left with a large chunk of untagged spend. Further, there is some spending that cannot be tagged at all, or it can be tagged but is not subsequently reflected in billing data. Given these two challenges, a hierarchy-based account strategy often offers the cleanest chargeback options.

Organizing Accounts and Projects into Groups

The big three cloud providers offer ways to group and organize your account, project, and subscriptions into hierarchies via AWS Organizations, Azure management groups, and Google Cloud folders.

Here is a sample of how grouping works according to Google's documentation for folders:

> A folder can contain projects, other folders, or a combination of both. Organizations can use folders to group projects under the organization node in a hierarchy. For example, your organization might contain multiple departments, each with its own set of Google Cloud resources. Folders allow you to group these resources on a per-department basis. Folders are used to group resources that share common IAM policies. While a folder can contain multiple folders or resources, a given folder or resource can have exactly one parent.
>
> In [Figure 12-2], the organization *Company* has folders representing two departments, *Dept X* and *Dept Y*, and a folder, *Shared Infrastructure*, for items that might be common to both departments. Under *Dept Y*, they have organized into two teams, and within the team folders, they further organize by products. The folder for *Product 1* further contains three projects, each with the resources needed for the project. This

provides them with a high degree of flexibility in assigning IAM policies and Organization Policies at the right level of granularity.

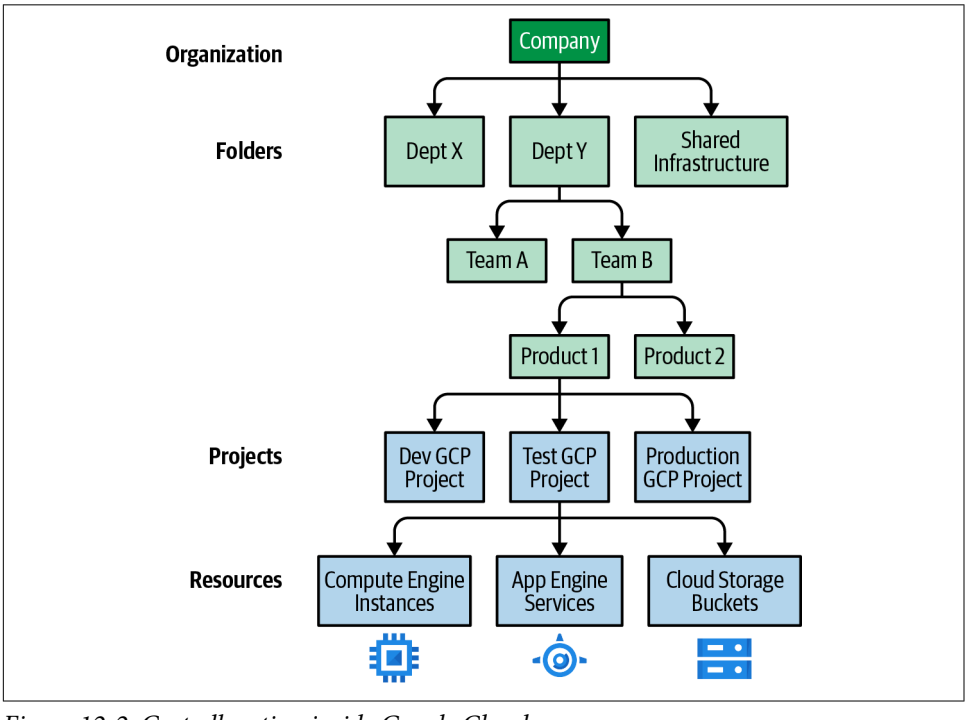


Figure 12-2. Cost allocation inside Google Cloud

It's quite common for a company to create folders that in turn contain additional folders or projects, as shown in Figure 12-2. This structure is referred to as a *folder hierarchy*. Taking a step back, the progression from organization to folder(s) to projects is referred to as the *resource hierarchy*.

Tags and Labels: The Most Flexible Allocation Option

Cloud service providers may call them different things, but fundamentally tags and labels both provide functionality to define metadata in the form of key/value pairs. These tags are then associated with the resources in a cloud account. The key is like the column heading on a spreadsheet, and a value is a row entry for that column.

Think of key/value pairs in terms of describing a bunch of shirts. Each shirt might have a tag key of *color*, which has a tag value of *red, blue*, or *green*. If you wanted to find the cost difference between red and blue shirts, you could use the *color* tag to group your costs and show the difference.

Figure 12-3 shows some cloud resources tagged with an *Environment* and a *Tier* tag. These tags provide a method for identifying which resources are from production versus development and which resources support the frontend as opposed to the backend of a service. To the cloud service provider, tags are merely strings of characters with no semantic meaning. Because tags are meaningful only to the customer, they can be whatever you want them to be as long as they adhere to syntactical limitations such as character set and length. A clear plan for tag keys and their appropriate/approved values will ensure that you do not end up with inconsistently applied tags that will hamper future cost allocation efforts.

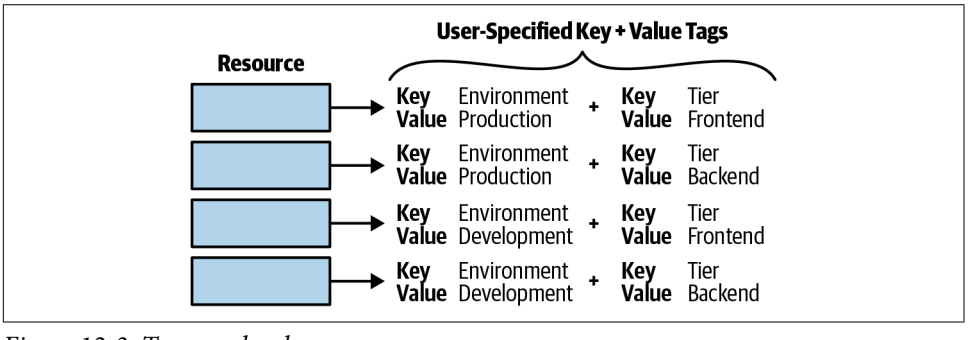


Figure 12-3. Tags on cloud resources

Using Tags for Billing

Having tag data represented in your cloud bill allows you to group the costs by multiple tag dimensions. Arranging costs by each of your teams—or grouping by different environments, such as development versus production costs—will enable you to allocate expenses to the right area of your business.

Tags allow you to identify which items on your cloud bill are attributable to each of your business's services. In addition to grouping your costs by these tag dimensions (teams/services/environments), you're able to monitor each group for significant changes in cloud billing—and therefore to address unexpected costs more quickly.

Some cloud service providers need you to select ahead of time the tag keys that you would like to have exported into your billing data. Others export every tag associated with your resources.

For the platforms that allow you to pick and choose your tags (like AWS), we caution against going crazy and adding hundreds of tags. Extra, unnecessary data in your billing files can increase the—already large—data size and make analytics harder.

Getting Started Early with Tagging

As you begin to build out your FinOps practice, it's tempting to delay the hard challenges, like getting teams to tag their resources. Often companies either do not have a tagging standard or have one but don't enforce it with their teams. It's imperative to realize that tags aren't applied retroactively. You must plan your tagging strategies ahead of time to ensure comprehensive reporting.

For example, creating a resource on January 1 without assigning a tag to it until February 1 leaves all of the January billing data untagged, and therefore potentially unallocated. Also, tagging resources but not activating the tag in billing until a later date leads to gaps in your tagging allocations. Later, when you need to analyze billing data for the cost allocation and cost optimization strategies we'll discuss later in this book, you will find that the billing data does not have the allocation history that you need.

Tagging is a vital early step that ensures you'll have a rich dataset to work with when you get into more advanced stages of your FinOps practice. If it hasn't been said enough: don't delay your tagging strategy.

Deciding When to Set Your Tagging Standard

The best time to implement a tagging standard is when a company is just beginning its cloud journey. That standard should clearly define tag keys—and the appropriate values the tags may contain—that will be applied to all resources. Implementing this standard and communicating it to engineers before they start building will help ensure good tag coverage and will provide quality data. However, many companies have already built out large deployments and have no standard on tagging. Never fear: it is not too late to apply a standard and to have your engineers update their resources, it's just going to take some work.

It's not uncommon for people to resist implementing a tagging standard, especially when a large number of resources have already been deployed that don't currently meet the new tagging standard. Companies with leadership buy-in for the new tagging standard have better adoption of tags across the organization. Leverage leadership mandates initially—and ultimately cultural changes—making tagging an essential requirement not only for FinOps teams but to the business at large to encourage widespread compliance.

The value of tags *does* extend outside the scope of FinOps, and understanding these benefits can be useful when a business is trying to implement a tagging standard. For the security team, tags can identify the service and team responsible for resources affected by a possible security issue. Operations teams are easily able to determine what services will be impacted by a notification of a resource having issues. A good tagging policy in an organization will pay dividends for many different stakeholders.

The more thought-out a tagging standard is at the start, the less likely a future revision will be required. If you need to go back to a team and ask them to change a bunch of tags after they just finished rolling out your previous strategy, you're not going to make any friends.

As a business scales out on the cloud, it will be generating thousands, if not millions, of resource charges, each with its own tags. Changing a tagging strategy means that teams will have to update all of these resources. As tags are commonly applied by automation infrastructure like Terraform, CloudFormation, and Azure Resource Manager (ARM) templates, updates to automated tags will need to be rolled out in code. Companies often find that many resources cannot be updated, and they end up having half-implemented standards.

Picking the Right Number of Tags

A tagging standard shouldn't force teams to apply too many tags. Requirements for an excessive number of tags will only cause resistance to the standard, which will lead to noncompliance. Always keep in mind the questions that you need to answer, and balance your tag strategy against the data you can get from your account hierarchy. It's best to aim to require tags only for the core metadata needed. Instead of defining all tags as required, define optional tags that clearly describe how your teams should implement them if they choose to do so.

If you're unsure of where to start with a tagging standard, here are some tags that companies with successful FinOps implement:

- A *cost center/business unit* tag that clearly defines where the costs of the resource should be allocated within your organization.
- A *service/workload name* tag that identifies which business service the resource belongs to, allowing the organization to differentiate costs among the services that teams are running.
- A *resource owner* tag to help identify the individual/team responsible for the resource.

- A *name* tag to help identify the resource using a friendlier identifier than the one given to you by your cloud services provider.
- An *environment* tag to help determine cost differences among development, test/staging, and production.

We recommend using an account hierarchy in place of the most important one or two tags—say, cost center and environment—and then using tags for the remaining three to five allocations that must be tracked. A coarse-grained account hierarchy provides for easy mapping to a few dimensions, with tags providing the finer details of additional business dimensions.

The most mature FinOps practices use tags to connect cloud resources to other sources of information, such as configuration management databases (CMDBs). This both enables large amounts of data to be referenced and allows quicker/simpler processes of updating information once and not having to correct potentially millions of tags on cloud resources.

Working Within Tag/Label Restrictions

It's best to ensure that your planned tagging policy (of required keys and values) will work across all cloud service providers you plan on using. Some cloud service providers' services allow almost any character, while others are stricter. Tags can also have limits on their length, and services have restrictions on the number of tags supported per resource.

You should ensure that the services you intend to use will work with your proposed policy. Otherwise, you may end up having to change your tags at some point, or you may end up with multiple groupings with slightly different values.

For example, you may start tagging your resources with *R&D*, but then realize later that a particular service does not support the & character. This leaves you with two options: to go back and retag all the existing *R&D* resources as something like *RandD*, or to end up having both *R&D* and *RandD* values in your bill and your costs split across the two groupings. Some FinOps tooling offers ways to fix inconsistent tags after the fact by allowing you to apply rules to adjust how resources are allocated in their reports, without requiring changes to the raw cloud spend data.

Some cloud service providers allow you to tag more of their resource types than others do, which can be a common complaint from staff members tasked with tagging resources. The important thing here is to realize the benefit for all the resources that support tagging, and to continually push your cloud service providers to add more tag support to their resource types.

Even if you can't tag all of the resources, you should still make an effort to tag as many as you can. There are plenty of business benefits from applying tags to the resources that support it. Untaggable resources, like CloudWatch fees, are one reason we recommend using some form of hierarchy grouping in addition to tags. Then you can allocate as many of the untagged costs as possible.

Maintaining Tag Hygiene

Tag hygiene refers to the actions you take to ensure your tag system (and the data that comes from it) is as clean as possible. Without proper tag hygiene, you risk working with inaccurate data, which in turn will throw off the accuracy of your decisions.

Consistency is key when it comes to tag hygiene. If one team uses a tag value of *prod* and another uses *production*, these tags will, by default, group as separate line items in your billing data. Tags are sometimes case-sensitive, so the value *Enterprise* may be reported separately from the value *enterprise*. When human beings create tags, it's common to see misspellings, variations in case, and abbreviations. The answer here is to enforce tagging policy through automation. Infrastructure automation, such as Ansible, Terraform, Azure Resource Manager templates, Google Cloud Deployment Manager, and AWS CloudFormation, can help ensure consistency and avoid human error.

However, even with the best automation and deployment practices implemented inside an organization, it's inevitable that some resources might still get created with either missing or invalid tags. That's why there are open source tools and third-party platforms designed to remove or at least stop these resources close to the time they are created. This helps limit the amount of unallocated/unexplained costs that will otherwise be generated inside accounts. (We will cover automation during the operate phase of the FinOps lifecycle; see Part IV.)

Another effective option is to replicate tags down from parent resources (such as an instance) to the child resources (e.g., volumes, snapshots, network interfaces). By copying down the tags to the child resources, you get greater tag coverage and can be sure the resources are getting tagged with relevant tag sets. We've seen successful tagging strategies that require tagging only of Azure resource groups, for example, and not of the resources themselves, while others tag both levels and implement business logic to decide which value to use in specific circumstances.

You can even create some business logic to allocate untagged resources to default cost centers. One example of this is to apply a default cost center to each of your cloud accounts. All untagged costs within an account will then be allocated to the assigned default cost center. For billing, you can do this at the resources themselves by updating the tags. Or you could potentially apply this business logic to the billing files after the fact, during analytics. Applying this logic after the fact also gives the

benefit of being able to update your business logic without having to go through your accounts and update actual tag sets on individual resources.

To explore tooling that can help with cost allocation, tag hygiene, and applying logic to cost reporting, check out the FinOps Certified Platforms on *FinOps.org*.

Reporting on Tag Performance

To determine how your tagging practices are working inside your organization, you should have reports identifying where tags are incorrectly applied. We recommend you measure tag coverage by the amount of cloud spend with a tag allocated versus without. Measured in this way, very short-lived, low-cost items are reported as less important than long-running, high-cost resources. Remember, not every resource is taggable inside the cloud service providers, so it's more realistic to set a target of 95% and work out a process to allocate the last 5% untaggable resources.

Reporting allows you to follow up with the relevant teams to update their automation—or at least to manually correct the tags. By tracking how many resources aren't matching your tagging standard over time, you can determine whether your organization is getting better at applying the tagging standard. Having a worst offenders list that everyone can see also helps to drive more thorough tagging adoption.

Getting Teams to Implement Tags

One of the common questions we hear is, "What are your top points of advice for getting resources tagged?" Again, we turn to Ally Anderson, whose story we shared in the last chapter. She works closely with the engineering contacts in each team, and says, "If there's a large amount of spend coming untagged, I make it visible to them via a report." She found that the more consistently she gets the data in front of the team leader, the more tags get applied, simply by creating a real-time data feedback loop. She also gets reinforcement from her leadership team. As part of her executive reporting, Anderson has made untagged spending a core part of ongoing reports, which encourages executives to push their own teams to minimize untagged costs.

Darek Gajewski from Ancestry.com focuses on a tag-on-create approach as a strategy for increasing tag compliance. Since cloud providers keep adding more and more support to require certain tags to be set in order to launch infrastructure, taking advantage of that can be a huge help in increasing tag coverage.

To hear all the concepts in this chapter put together into a real life story, listen to Episode 6 of the FinOpsPod podcast called "Attribution, Tags & Labels . . . OH MY!" which tells Jason Rhoades's story of how Intuit started, evolved, and matured their attribution strategy over years of iteration on tags, labels, and account structures.

Conclusion

Good FinOps practices should start with a carefully crafted account and tagging strategy. The extra metadata that account hierarchy and tagging bring to your billing data will play a part in almost all of your reporting and analytics going forward.

To summarize:

- Your account allocation strategies will provide structure to the resources and help you group costs.
- Having your company understand the importance of the tags and work on automating them will benefit the whole business.
- Cost visibility and cost accountability built from your tags and account allocation will assist in building a lean culture within your organization.
- Tagging and account allocation is a common requirement throughout the optimize phase of the FinOps lifecycle.

Formulate a cost allocation strategy first—it will help with almost all of the FinOps capabilities, especially forecasting, which we cover in the next chapter.

CHAPTER 13
Accurate Forecasting

Forecasting is the practice of predicting future spending, usually based on a combination of historical spending trends and an evaluation of future plans. It is done to understand how future cloud infrastructure and application lifecycle changes may impact budgets and to aid in evaluating cloud investment decisions. And it's one of the hardest things to get right in FinOps.

Forecasting is hard because it requires an intersection between engineers, finance, and procurement, where finance has financial reporting responsibilities, procurement has accounting responsibilities, and both need assistance from engineers and executives to meet these obligations and understand planned changes to infrastructure. Collaboration between these stakeholder teams is critical to build agreed-upon forecast models and KPIs from which to establish budgets that align with business goals. When these teams build models to forecast cloud spend reliably and accurately, cloud cost forecasting will inform investment and operational decisions to accelerate an organization's growth.

In this chapter, we discuss forecasting, explain why forecasting is so difficult, and go over the key concepts needed to improve its accuracy.

Traditional IT forecasting—prior to cloud—was mostly done for annual budget processes. For equipment in the data center, forecasts for the total cost of ownership (TCO) are calculated up front when performing system enhancements or net new system builds. As engineering teams have historically not been involved or concerned with forecasts on IT spending outside of building the business cases for capital spend, they do not have the muscle memory of frequent forecasting and collaboration with finance and business.

In a traditional data center deployment there is a large percentage of known fixed costs, so forecasting inaccuracies are likely to be limited in impact. However, when

you move to the variable cost model of cloud, the amount of fixed versus variable costs is flipped, with the greater being variable costs, and inaccuracies in forecast can be material.

Frequent forecasting in cloud is much more important than in a data center deployment and will require more interaction among everyone.

The State of Cloud Forecasting

Forecasting of cloud spend can be a challenging area to master even for advanced practitioners. It was the second most common FinOps challenge in the latest State of FinOps survey, and a large segment of respondents—including those from mature practices—reported variances of over 10% on their forecast accuracy. Ten percent can add up quickly when your cloud bill reaches into millions of dollars per month.

Variances against forecasts stem from cloud cost complexity combined with the distributed consumption of cloud resources: when a lot of disparate people and code are procuring infrastructure as needed, it's really hard to nail down what it will all add up to in 6–12 months. It's not only a modeling problem, but also a people and technology one.

The State of FinOps 2022 respondents show there has been some improvement with benchmarking, showing that forecasting has become more accurate compared to responses in the 2021 survey. The most advanced stage cohorts were able to get their forecasts within 5% variance of actual spend, whereas the less advanced stage cohorts reported variance of 20% or more. Accurate forecasting is time-consuming and requires automation and mastery of other FinOps practices to increase efficiency.

Advanced stage respondents led the way in having faster cycles in refreshing their forecasts and related cloud budgets, although a minority of respondents generate forecasts at a weekly cadence, indicating much room for improvement for all. About 21% of respondents report forecasting annually, which is a surprise considering the variability and speed with which cloud infrastructure can change throughout a year, a likely cause for forecasting accuracy appearing in the top challenges for FinOps teams.

> *It's going to be wrong. Get over it and start with an educated guess, but do it fast!*
> —Andrew Feig, Managing Director, Cloud Engineering at JPMorgan Chase & Co.

A large number of survey respondents stated that they do *no forecasting*, leaving them flying blind until the bill arrives, or simply that they are not widely communicating their forecasts due to the lack of accuracy.

Forecasting Methodologies

Let's take a look at the methodologies used by organizations. It is important to cover these basics first, since your organization may have a different name for each of these methods. Clarifying which is which will help us reach a common understanding.

The first set of terms to align on is the method of *how to generate the forecast* itself:

Naive forecasting
: A very simple method that uses your previous spend history. Naive forecasts make the assumption that your next period of cloud spend will be the same as your previous. In Figure 13-1, the forecast is a direct copy of the historic actuals. This method of forecasting is too basic for cloud spend, which can change quickly based on demand, business events, and human choices.

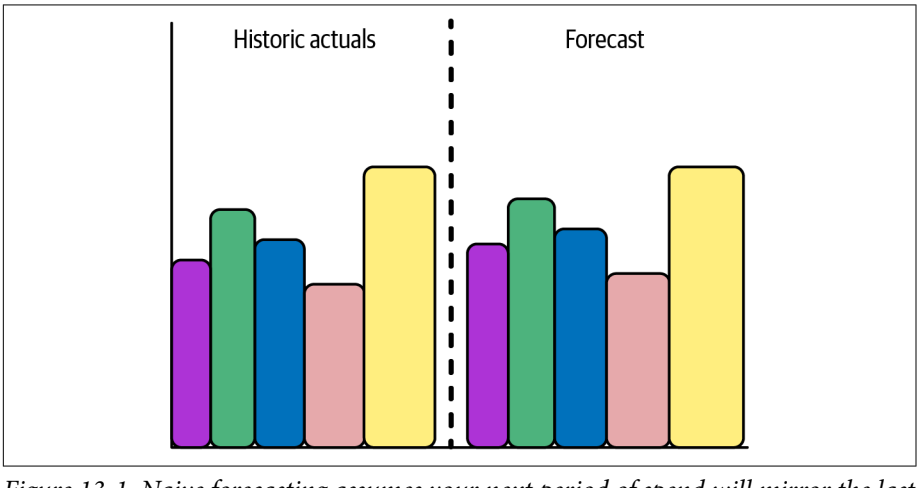


Figure 13-1. Naive forecasting assumes your next period of spend will mirror the last

Trend-based forecasting
: Also known as *univariate forecasting*, this method still uses past cloud spend history to base the forecast on, but it also introduces some form of statistical element—such as linear regression—to use the trends of past forecasting to predict the spend in future periods. In Figure 13-2, the historic trend of cloud spend is used to predict the future periods. Trend-based forecasting considers only your existing historical spend and does not introduce any additional scenarios or business drivers that may change in the future period.

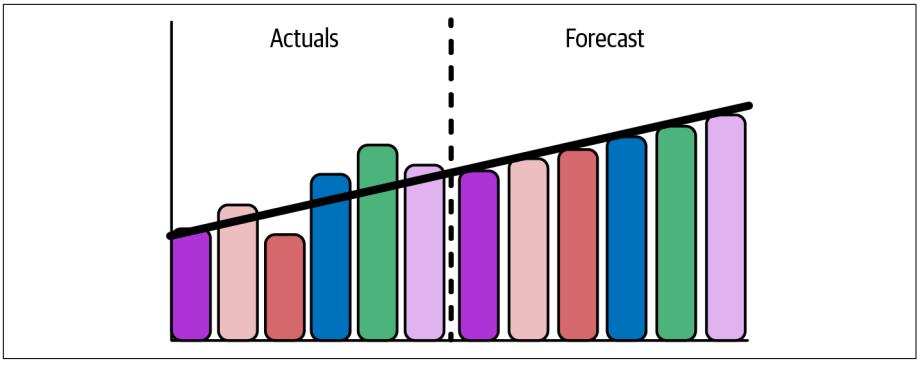


Figure 13-2. Trend-based forecasting

Driver-based forecasting

Also known as *multivariate forecasting*, this method considers other business metrics to augment your forecast. Connecting a business driver for your cloud spend can increase the accuracy of your forecasts, especially when this other business metric has already been forecast with an acceptable level of accuracy. This can be in the form of manually adjusting the forecast or having multiple input data sources to an automated forecasting algorithm. In Figure 13-3, the trend-based forecast is adjusted for additional expected spend due to external drivers compared to previous cloud spend. Some examples of this include:

- Increased activity from a marketing initiative
- Changes in cloud usage due to a new product release
- Staff headcount compared to research and development (R&D) cloud spend
- Sales/monthly active users compared to production (COGS) cloud spend

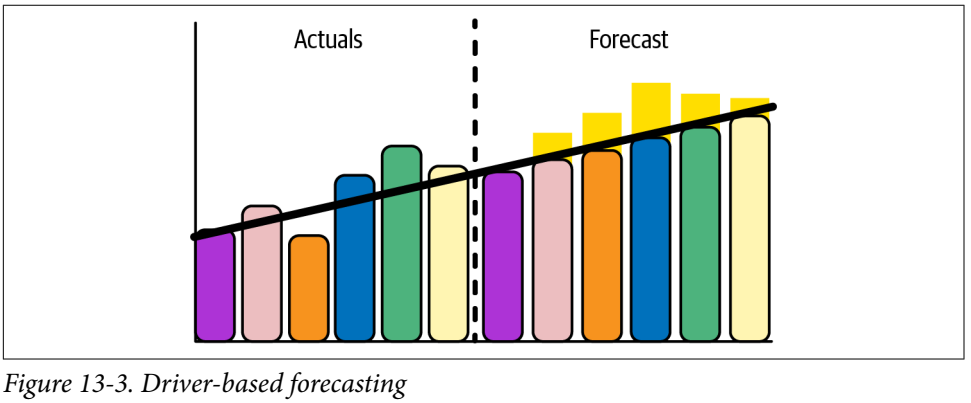


Figure 13-3. Driver-based forecasting

Forecasting Models

Now let's consider the tools or models you might use to generate a forecast. We won't go into details about different algorithms or machine learning models that generate forecasts. The aim of this chapter is to assist with the way you think about your cloud forecasts and consider techniques to help you work with cloud spend forecasts within FinOps. If learning specifics on forecasting interests you, there are many good books that cover forecasting such as *Forecasting: Principles and Practice* by Rob J. Hyndman and George Athanasopoulos (OTexts, 2021). Forecasting models include:

Machine learning forecasts
: This is a term that is often overused to describe forecast technologies. Some people consider only deep learning models to be truly machine learning forecasts; others include statistical and other nondeep learning models in this category. No matter which model you choose to use, we recommend you shop around. Don't be afraid to try another model. The first one you use is unlikely to be the best. Also consider evaluating multiple models against history to see which ones would have been most accurate.

Composite forecasts
: This is an approach that combines different forecasts to achieve more accurate predictions using more than one model. A simple method for combining is to average the results given from multiple models.

Static forecast
: This type of forecast is generated once at the beginning of a set period, such as 12 months. It is not updated until the period ends, when a new forecast is generated for the next period. This type of forecasting is good for very predictable spending.

Rolling forecast
: This type of forecast is also generated at the beginning of a set period, such as 12 months. However, unlike static forecasts, it is regenerated frequently (weekly, monthly, quarterly) during the period. Each time it is regenerated it is created for the same length of time, effectively adding more data to the end of the forecast horizon.

Cloud Forecasting Challenges

To generate more accurate forecasts, you must first work through a set of common challenges. You are likely working on some of these items already, as the capabilities needed to solve for forecasting relate to many other items in the FinOps Foundation Framework covered in Chapter 7. Depending on the structure of your organization

and how well established FinOps is, some of these challenges may be easier to solve than others.

Manual Versus Automated Forecasts

Forecasts often start out as a manual exercise for the finance or FinOps teams within an organization. The exercise may use either a naive approach of using previous spend numbers and then adjusting with engineering feedback, or moving historical data into a spreadsheet and using formulas to generate some trend-based forecasts. As your cloud spend scales, this manual approach becomes difficult, and accuracy often suffers. Costs from low-spending teams are often grouped into a single group and forecast as one, reducing the granularity of the forecast. The effort to generate these forecasts means the frequency is very low—monthly, quarterly, or less.

Automated forecasting tooling that can perform trend-based or driver-based forecasting enables forecasts to be generated often. In the State of FinOps survey 2022, the most advanced practitioners are reportedly executing forecasts weekly or even daily. A mature FinOps culture will lead to more engineering teams taking responsibility for their spending, through which they become more interested in forecasting.

Inaccuracies

Look for models/tools that can take into account the seasonality of your cloud spend. It is unlikely that your cloud spend is the same all year round. Having a forecast that can factor in how your cloud spend ebbs and flows will result in much higher accuracy.

Some models are affected more by outliers in the data; a short, sudden spike created by an anomaly can cause forecasts to be inaccurate. Some models handle outliers well, while others will require you to make manual adjustments to fix them.

A single algorithm isn't enough. Some forecasting algorithms work better with a longer history of cost data, whereas others are able to work with limited data points. It's likely you will have applications with a long history of data, but when you introduce new workloads, most forecasting algorithms won't work as effectively. An algorithmic ensemble approach that incorporates multiple scenarios is a much more effective way to accommodate the variances in availability of history.

Granularity

Forecasting on a total cloud spend level does not allow you to figure out what individual groups of cloud spend are driving the inaccuracy of your forecasts. In addition, your finance team will likely need a finer granularity of forecast for their

reporting requirements. Conversely, generating forecasts at a very detailed level can take more effort than may be warranted, especially when you have large numbers of groupings with very small amounts of cloud spend.

Chapter 12 covered tagging and labeling, without which generating forecasts with any granularity is almost impossible. Tag- and hierarchy-based approaches allow you to divide the cloud spend into relevant categories like cost center, application, or environment. Forecasting at finer levels of granularity can enable you to determine where your forecasts are not matching actuals and allow you to focus on these areas to improve.

When generating granular forecasts, it's expected that the forecast is coherent with a summary forecast. Using the spend hierarchy in Figure 13-4, if you generate a per application forecast, it's expected that adding these together would equal your total cloud spend forecast. However, if you generate a summary forecast for your total cloud spend in addition to a granular forecast, such as at the application level, it is likely—due to the statistical properties of forecasting algorithms—that adding up all the granular forecasts will not equal the total of an independently generated summary forecast. The difference between the forecasts will be driven by small inaccuracies at the application level for the granular forecast and by the lack of detail available in a summary forecast.

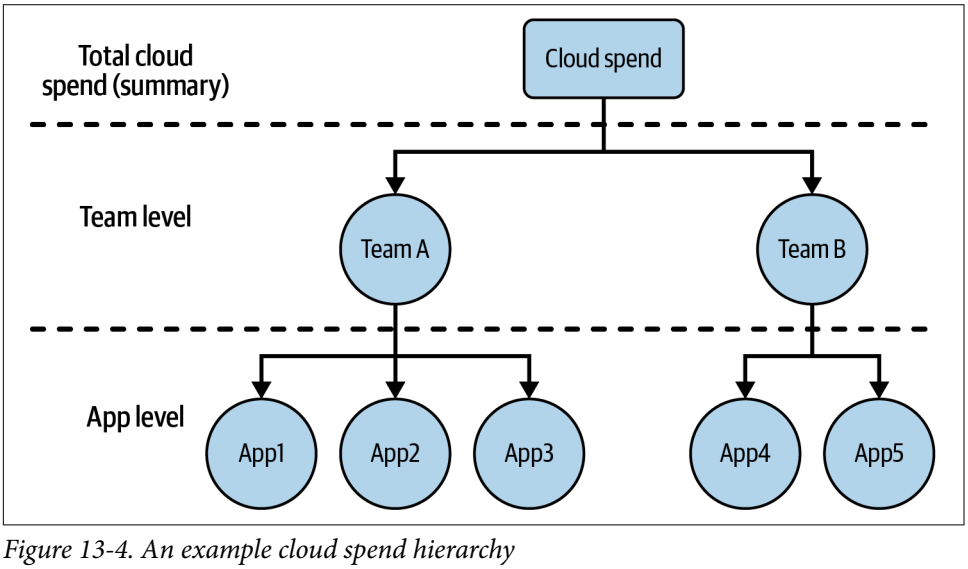


Figure 13-4. An example cloud spend hierarchy

You can create coherent forecasts by limiting your projection methodologies to one level of granularity. Using the layers in Figure 13-4 as an example, there are three main approaches to forecast granularity that most forecasters use:

Bottom-up
: The bottom-up approach focuses on the most granular level of cloud spend; our example uses application-level spend. This approach creates individual forecasts for each application then summarizes it into broader classifications, teams, and then total cloud spend. This approach allows for significant inputs from engineers and executives, which increases accuracy at the granular level and may lead to the most accurate total spend forecast.

Top-down
: The top-down approach focuses on generating a forecast for total cloud spend using both historical spend trends and future project information. This forecast then gets allocated to the multiple teams and applications using historical proportions matched with future project expectations. This approach is most successful when there is little to no variation in the distribution of spend across applications, but it becomes highly inaccurate in cases where there are large fluctuations in spend distribution.

Middle-out
: A final approach to generating coherent forecasts is middle-out. In the Figure 13-4 example, this would entail generating a forecast at the team level and then summing up to the total cloud spend level and proportioning down to the application level.

Generate a forecast for the granular level that is most important for your reporting. Apply a middle-out approach to summarize the forecast and then, when needed, apply historic proportions to expand the forecast to more granular levels.

Forecast Frequency

Remember the cloud moves fast, and the faster it moves the quicker your forecasts are going to be out of date. Imagine one of your engineers making a decision today that increases their cloud spend by $10 a day. Over the next 12 months this decision will accumulate to over a $3,500 increase in spending. Now think of how many engineers you have making $10 decisions every day of the year. A 12-month static forecast today will be wildly out of date by the latter half of the year.

The most advanced practitioners in the State of FinOps reported updating their forecasts weekly, though monthly or quarterly was still the most common cadence for the majority. For forecasting accuracy, that is simply too long to be effective.

When you recalculate your forecasts more frequently using a rolling forecast, you maintain a more accurate vision of what your cloud spend will be; in addition, you can use this information to find what is driving the forecast changes. It is much easier

and quicker for an engineering team leader to look at their spend for the last week, spot any discrepancies from what they expected, and correct next week's forecast, than to do this monthly or quarterly.

The data in Table 13-1 shows a forecast for five applications. This forecast is for a monthly spend for each application and is generated weekly. By comparing the difference between the previous and current forecast, you can easily identify where the forecasts are changing and see that App2 is the team you should ask for more information to understand the change in forecast more accurately.

Table 13-1. Forecast changes week over week

Application	Monthly forecast (week 1)	Monthly forecast (week 2)	Difference
App1	$15,000	$15,000	$0
App2	$25,000	$30,000	+$5,000
App3	$10,000	$9,000	–$1,000
App4	$5,000	$6,000	+$1,000
App5	$30,000	$30,000	$0
Total	$85,000	$90,000	$5,000

> *None of us is as smart as all of us.*
>
> —Ken Blanchard

As your engineers across your organization are closer to the causes of cost variances, it's easier—and quicker—for them to identify cost drivers. A single small variance may seem immaterial compared to your total cloud spend but might be significant to a single team, and many small variances can add up to material amounts. Providing your engineers with granular cost visibility and regularly updated forecasts enables you to get the biggest benefits.

Communication

We talk a lot about enabling the business to make decisions based on the value of cloud, and here you need these decisions to make their way back to the forecasting process to be captured and estimated correctly. No amount of machine-driven forecasting is going to accurately predict the changes your engineers have planned for their infrastructure and how that will affect your cloud spend. In our previous section we gave the example of engineers making small decisions on cloud spend and how these add together to make your organization's cloud forecasts inaccurate. It's required that there is an open communication channel that captures these plans from engineers when generating forecasts. It is important for you to treat forecasts as more than an internal finance effort. Forecasting needs to engage all of the FinOps personas and leverage the expertise of each persona to create the most accurate forecast possible.

Future Projects

Future projects can range in impact from small cost decisions that have minimal impact on the total cloud spend to large cost decisions that can massively change the way the business utilizes and spends on cloud. For example, migrating a large application from data centers into the cloud or deploying applications to multiple new regions to offer your products in more locations are large-scale business changes that can impact cloud spend forecasts. These large-scale decisions, when made, often add net new cloud spend to your actuals, which makes it impossible for any machine-driven forecasting to predict the future increase. Establish collaboration processes so that your engineering teams have defined ways to involve your FinOps team in estimating the cost impact of future projects. This will ensure that the extra spend can be manually entered to augment the forecasts and improve their accuracy. You can see in Table 13-2 the automated forecasts for applications and the extra manually entered project costs to create the complete forecast.

Table 13-2. Forecast changes week over week

Forecast type	Application	Forecast Month 1	Forecast Month 2	Forecast Month 3
Automated	App1	$15,000	$16,000	$17,000
	App2	$25,000	$30,000	$35,000
	App3	$10,000	$8,000	$6,000
	App4	$5,000	$5,000	$5,000
	App5	$30,000	$40,000	$50,000
Manual	Project A	$0	$25,000	$35,000
	Project B	$0	$0	$10,000
	Total	$85,000	$124,000	$158,000

One item to be cautious of is the estimation made by engineers on when extra cloud spend from future projects will be deployed. Timing is often overly optimistic and can land much later than first predicted. Gathering this information from engineers should be continuous, updating the forecast until the project lands and slowly refining the timing of the new spend with the most up-to-date predictions available.

Stories from the Cloud—Bharat Chadha

Bharat Chadha manages IT operations at Freewheel. His forecasting journey started out using trend-based forecasting utilizing statistical formulas without engaging engineers. He started to see large discrepancies, so he decided to change to a bottom-up approach using a monthly survey to the account owners to validate the automatically generated forecasts.

Using business drivers to correlate revenue to increases in AWS cost, he was able to adjust cloud budgets. He worked with the CTO office to get support and then reached out to the approximately 90 account owners to meet every two weeks to review forecasts. Attendance is high, and they focus on the main drivers of variance. The account owners are aware of the forecasting formulas used. They have been able to achieve 82%–85% forecast accuracy, which is sufficient for Finance but still leaves room for improvement over time.

His takeaways: "It really helps to see trends and map events to specific months. You need good training around tagging, e.g., when hiring engineers they need to know about tagging and how important it is. Culture is hard to change; you need to set expectations at the beginning."

Cost Estimation

Predicting the cost of a future deployment can be very difficult for engineering teams, especially if there is no existing proof-of-concept work or vendor-supplied calculations for the application deployment.

Cost calculators made available by all the major cloud service providers help get a ballpark value for the new workload, but these tools are only able to estimate the costs of the items you manually enter into the calculator. Due to the complexity of cloud deployments and the granularity of costs, it's easy for your engineers to miss entering items or underestimate the size/volume of the resources needed to operate their new workload.

Comparing existing workloads that are similar in design to the new workload being introduced can help teams estimate the future cost of the new workload. Have your engineers compare the design of the new workload being introduced to existing workloads to identify if you have any analogous systems. Where possible, using existing workloads will lead to identifying cost items that could be missed when using cost calculators. If there is no existing system, use the variable cost model of the cloud to your advantage; have your teams deploy the new workload in a test environment and operate it for an hour or so. After this test you can then identify the costs associated with the test run of the workload.

Some of the most difficult costs for your engineers to estimate prior to deployment include the following:

- Network traffic costs
  - — Data transfers out of region
  - — Cross-zone transfers
  - — Volume of traffic handled by individual resources like load balancers

- Number of events
  - — Function as a Service executions
  - — Messages sent/received on queues
  - — Number of database queries
- Activity of data storage
  - — How much data can be transitioned to less frequently accessed tiers
  - — Count of requests for the individual objects

In the vendor and open source space we are starting to see new tools like Infracost, which enables your engineering teams to see an estimate of the cost impact of their changes before they roll them out to the cloud platform. This kind of shifting left provides your teams the ability to check that their changes will have the expected impacts to costs.

Whether it's before your engineers deploy new infrastructure or before they deploy an update to existing infrastructure, frequently updating forecasts and the communication of project work means your forecasts will be refined as your engineers progress with their projects. Collaboration between engineering teams enables you to capture feedback on what is changing in forecasts as the updates roll in.

Impacts of Cost Optimization on Forecasts

Just as business decisions and new deployments can drive up cloud spend, future cloud cost optimizations can greatly reduce spending, also making your forecasts inaccurate. The chapters in Part III cover these optimizations in more detail. For now, we want you to understand that just like project work, which contributes to increased cloud spend, you need to generate predicted efficiency calculations that represent the reduction in cloud spending driven through FinOps activities and subtract them from your forecasts.

In the early stages of implementing FinOps, there can be a drastic reduction in cloud spending as you introduce usage and rate optimizations. Without predicting these reductions within your forecasts, you may end up forecasting above actuals.

When you introduce FinOps optimizations, the reduction of your operating costs may hide the natural growth rate of your cloud spend. As these optimizations stabilize, the normal growth rate of spending will return; it's important to model this when tuning your forecasts so the initial reductions don't cause you to underestimate things in the future.

An example of how FinOps optimizations can impact forecasts when not taken into account can be seen in Figure 13-5. With FinOps activities starting in 2022-05, you see a rapid reduction in cloud spend. After a few months, in 2022-09, the natural

growth of the cloud spend returns. If a forecast was generated in 2022-04 without factoring in the expected impacts of introducing FinOps, the forecast would have been generated too high. If instead the forecast was generated in 2022-07, the downward trend would have impacted the forecast and the natural growth in cloud spending that is hidden under the cost reductions at the time; if they had not been accounted for, the forecast will be generated too low.

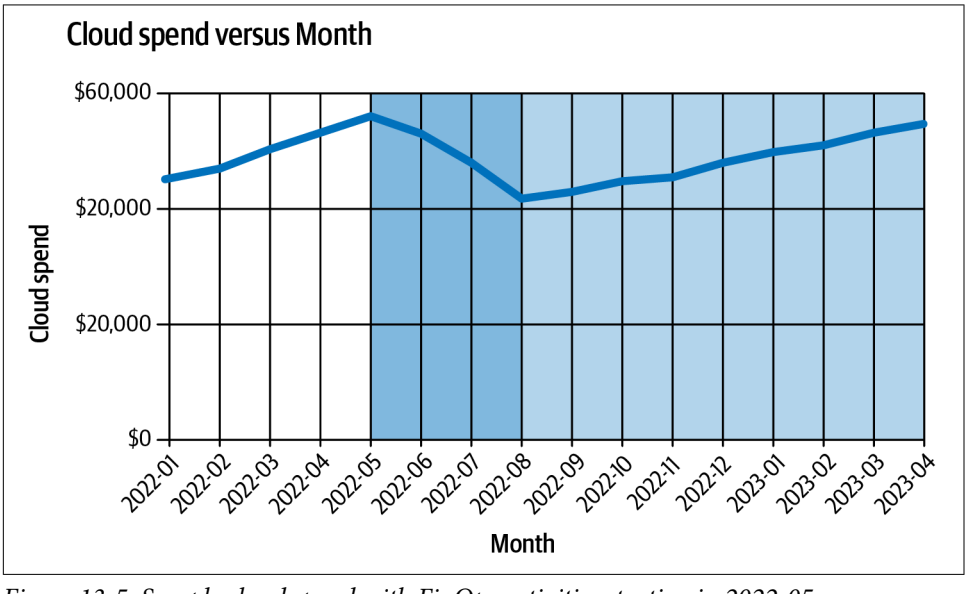


Figure 13-5. Sample cloud spend with FinOps activities starting in 2022-05

Stories from the Cloud—Drew Lowe

Drew Lowe leads the FinOps sourcing team for Hulu Disney, partnered with Anthony Logan. At the beginning his team had no mandate from the top, which allowed them to focus on policies. Financial planning and analysis (FP&A) was lacking technical depth, while engineers were not concerned with cloud cost. Hulu is primarily using AWS with some workloads in data centers. Drew's team ultimately reports to the CFO.

Forecasting reached a turning point when engineers asked for 4.5 times the previous year's spend for the next budget cycle. The decentralized engineering teams made pricing decisions on products without understanding the underlying costs. The company's objectives and key results (OKRs) put the viewer first. While Drew was able to connect spend to revenue, he was concerned about overspending.

A key OKR around accuracy to forecast was added for all engineers. Drew's team started with good data and made sure they could trust the numbers. By driving visibility, he was able to build demand-driven forecasting, which in turn allowed

them to optimize spend. By identifying the top-spending teams and learning about their workloads, his team was able to get the majority of the forecasts in line with expectations.

Drew is using a hybrid forecasting model, where he uses business driver-based forecasts for top services and trend-based forecasting for the long tail. This allows him to update forecasts every month with an average variance of about 2.5% and the worst case being about 9% variance. To get there, Drew decided to forecast by workloads instead of cloud stock keeping units (SKUs). However, engineering volatility is challenging, for example, predicting load testing or new services. For new services Drew uses swag estimates to track these special projects on their roadmap.

Drew's takeaway: "A tagging strategy is imperative. Partnering closely with the highest cost workloads works very well. Start building relationships with CTO and CFO, then engage with the engineering leads."

Forecast and Budgeting

Forecasting and budgets go hand in hand. We will cover the stages of budget generation soon, but it's important to understand that using forecasts to set budgets is one of the main contributors to having budgets that reflect reality.

If you're trending off of your forecast, you must ask yourself: does something need to be changed to bring you back in line with the forecast, or do you need to update your forecast? Therefore, you should track your actual spend versus your forecast every day. Graphs for cloud spend should include your forecast to determine if there are any actions you need to take (see Figure 13-6). Then you break down the information to the team level, which allows you to see whose actions, if any, are pushing you out of your forecast. There may be business reasons (compliance, performance, etc.) that mean you should make the choice to simply adjust your forecast.

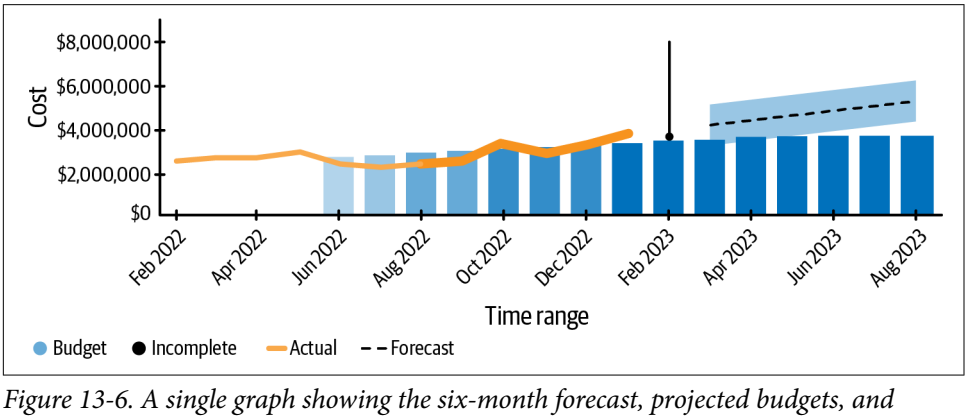


Figure 13-6. A single graph showing the six-month forecast, projected budgets, and actual budgets/use for the previous six months

The general consensus in the FinOps community is that cloud team forecasts should be done on a rolling 3- to 12-month basis. Atlassian does them in 3-month increments, but also projects forward 3 years to explore where they think they will be. A large media company we've worked with has a variation on this: a yearly forecast, broken into monthly targets that are revised every quarter.

Evaluating forecasts on a quarterly or monthly basis often isn't frequent enough. Skilled FinOps practitioners constantly evaluate variances based on live data (sometimes daily) to see which teams are deviating from quarterly forecasts and then lean in with them quickly. This lets them get in front of variances before they grow larger. When actuals vary from forecast and a decision is made to maintain this increase in spend, it is reported up the management chain the day a decision is made. Conversations can take place between management and the business to determine the value, if any, that the increased spend is generating and any remediations necessary.

These constant evaluations help the business make better decisions as the month progresses, while showing the soonest point the spend can be bent downward. Budgets shouldn't be meant to prevent spending, but rather provide a vehicle for identifying when spending deviates from plan—which, remember, enables conversations.

The Importance of Managing Teams to Budgets

It's easy to think of budgets as restrictive controls with harsh limits. However, budgets need not, and should not, restrict innovation. As we've mentioned before (and it's worth repeating!), FinOps isn't about saving money—it's about making money. In FinOps, budgets are about focusing your investment to fuel innovation instead of preventing it.

Initial conversations with engineering teams about cost optimization typically aren't productive. They haven't traditionally had to consider cost, and they are under constant pressure to deliver more and deliver it faster. So they are understandably resistant to spending their precious cycles optimizing for cost—often seen as someone else's job.

When a FinOps practitioner sits down with the architecture team to review findings on cost, the practitioner often encounters pushback on advice and only sometimes sees the suggested changes implemented.

However, if teams are given specific cost targets or goals to hit, they tend to hit them. Cost-savings advice from management is welcomed with open arms. Dev teams lean in, actively participating and asking for ways to cut their costs while maintaining speed and quality. Rightsizing happens because the teams have clear goals. A leadership-driven culture of cost optimization will recruit them as allies and give visibility into how they can help the business. They begin to understand that

FinOps is not about cutting resources but about enabling cost-conscious speed and innovation.

For many tech unicorns, budgeting is a four-letter word. It's a necessary evil that distracts from trying to gather up as much market share as possible in their highly competitive growth markets. They're immersed in a never-ending arms race to deliver bigger and better features, engaging content, or data-science-driven recommendations. There's an unfounded premise that budgeting slows it all down.

But for most enterprises, budgeting is a necessary step in launching a project. No green light is given until a budget is approved. The cloud has drastically changed the budgeting process. Gone are the days when a budget could be controlled by the power of a purchase order, which prevented paying anything beyond what was planned. Because of the distributed and lightly governed nature of cloud spend, most organizations—big and small—struggle to accurately budget in the public cloud.

Like most things, putting effective budgeting in place is a journey. Here are four levels of budgeting that organizations typically go through:

Level 1: No one is budgeting at all

During a recent meeting with a well-known tech unicorn, the team was asked how they defined budgets for their cloud spend. The answer was a shaking head: "We don't have budgets. We don't want to constrict engineers." This very same company had also realized they were spending considerably more than they wanted to. The organization had made a plan for how much they were going to spend in cloud for each of the next three years, but because it wasn't carried down to the individual teams responsible for consuming cloud resources, they were already well over what they and their cloud provider had agreed to. In this case they ignored the budget totally, not even practicing informed ignoring.

Level 2: Spend what you did last year

The budgeting process at a well-known retailer was as simple as trying to stay within what the company had spent the previous year—taking a static forecast and using it to set budgets. Though on the surface this seemed like a good starting point, it wasn't based on an analysis of whether what the company was spending was the right amount for what it was doing. Very much like government spending, it created the wrong incentives by encouraging teams to spend all of their budget so they wouldn't lose it the next year.

This type of budgeting—a "first stop the bleeding" approach—is actually fairly effective at halting out-of-control cloud spend. But it doesn't result in savings or any meaningful measure of cloud spend tied to business objectives. It's akin to how budgeting was done in the old world of CapEx-heavy data center purchasing. You looked at what you had then and how much capacity you had left, but you had no insight into the actual efficiency of what you were using.

Level 3: Cost takeout goal

Cost takeout goals look at the forecasted spend from a rolling forecast and then reduce it by a prescribed amount. They can provide the right incentives to teams, but they're often arbitrary and not tied to any specific business outcome. An online grocer had a goal to reduce spending across the organization and set a cost takeout target of 20% reduction for each of its teams. This target was fairly easy to hit, as it was the first optimization target the teams had been given. After all, "what gets measured gets improved." Machines were running that could be terminated completely, unused storage was lying around, and there were many opportunities to rightsize resources and purchase RIs. Over time, the cost takeout approach begins to be problematic for two reasons. First, savings in later periods are typically not as dramatic as those at the beginning, leading to questions about the value of FinOps itself; and second, the focus on total spend can start to drive cost cutting that impacts other aspects of system performance or value.

Level 4: Unit economic performance metrics

Here, companies begin to tie cloud spend to actual business outcomes. If your business is growing and you're scaling in cloud, it's not necessarily a bad thing that you're spending more money. This is especially true if you know the cost to service a customer and are continuously driving it down. As you know by now, tying spend metrics to business metrics is a key step to reach in the FinOps journey and is the direct connection to driver-based forecasting. Anyone can see whether a unit cost is going up or down, but they don't know why, so they can't tell you why. To truly enable distributed teams to affect their spending, activity-based costing is the final step. We'll discuss this in more detail in Chapter 26.

Ultimately, it's important that everyone with their hands on the wheel of spend understand how they are tracking, in-period, against their budgeted plan. This lets them know early enough what course corrections will be needed to hit the number if things are off. Overspending is not the only problem: to the finance team, underspending can be just as bad because it means that budget could have gone somewhere useful instead.

> *One of the things we continuously try to improve on is to eliminate forecast "placeholders" as more information is garnered. For estimates that are years out in the Finance Plan, "placeholders" are okay as technology solutions will have several options and not be locked down. As workloads get better defined, so can their forecasts, and placeholders should be evaluated and replaced with higher confidence estimates.*
>
> —Joshua Bauman, EA

To best steer the ship, engineering leadership needs to make the best possible trade-off for the organization on adherence to plan versus competing priorities. This means clearly communicating guidance on the strategic prioritization of the trade-off options and how they are expected to affect projected cloud spend figures.

Remember you can practice informed ignoring on budgets also. While most teams are expected to spend close to budget, some teams can be allowed higher levels of budget uncertainty. For teams developing new products or working on new strategic opportunities, you may allow more forgiveness for going a bit off plan, so long as the project goals they are aiming for are achieved.

Conclusion

Accurate forecasting is a challenge even for the most mature FinOps practices. Regardless of your methods, there should be an expectation that the closer the timeline, the more accuracy/specificity is expected. The next quarter should have much higher confidence than two years from now. There are a handful of changes you can make to the way you forecast cloud spend to give you the best chances of generating reliable forecasts.

To summarize:

- Don't use static forecasting; choose instead a rolling forecast model.
- Driver-based forecasting in conjunction with forecasts that can handle seasonality and anomalies give the best baseline forecast data.
- Granular forecasts are important for FinOps, but you will need to maintain coherence to summary forecasts.
- Transitioning from traditional IT forecasting to cloud spend forecasting involves getting all personas active in the forecasting process.
- Augment the machine-generated forecasts with feedback from both engineering and FinOps teams. Not every data point is available in your historical data.
- Budgets should be driven by forecasts and communicated down to the engineering teams regularly, with guidance on trade-off expectations between budget adherence and hitting other business objectives.

Now that you know what you expect to spend on cloud, let's dig into ways to optimize cloud spending in the next phase: *optimize*, the focus of Part III.

PART III

Optimize Phase

In this part, we walk through the processes to set and track goals for data-driven decision making that will optimize an organization's cloud. We also look at the cloud service providers' discount offerings and commitment strategies that can help to reduce cloud costs.

CHAPTER 14

Optimize Phase: Adjusting to Hit Goals

As you enter the optimize phase, your focus moves to making decisions in near real time, identifying anomalies, and spending efficiently. You use goal setting on metrics so your organization can understand how well it is performing against set expectations.

In this chapter we'll expand on why you need goals, how they should be implemented in the FinOps world, and how to set them. We'll also introduce the goals of cost optimization, which set the stage for the remainder of the optimize phase.

Why Do You Set Goals?

Every journey into the cloud tells a different story. For some, the speed with which all services are deployed into the cloud is more important than the cost of doing so. For others, the primary goal is to take it more slowly and ensure that the budget is maintained during the whole process. Others may already be in the cloud and are discovering that they're spending much more than expected. They understandably feel a need to get things under control.

It's worth noting that most organizations have multiple cloud stories going on at the team level. Some teams might be cloud native and looking to maintain their costs. Others are migrating or implementing new projects and are focused more on delivery time than on dollars spent.

By setting goals at the corporate level, and for each individual team, you're able to keep track of business decisions, set expectations, and identify where within your cloud spend things aren't working out as well as you hoped.

The First Goal Is Good Cost Allocation

Before you can set further goals, you must have a clear understanding of the current state. If you implement a clear tagging and account separation strategy as you scale out resources in the cloud, it'll be easier to make sense of what you are dealing with and how best to optimize it.

Every organization should be tracking spend by cost center/business unit. This not only enables each individual team to understand their impact on the overall cloud spend but also allows the business to determine which teams are driving any cost changes.

We've already covered how to implement a clear cost allocation strategy that helps identify spend by cost center. These cost allocation strategies need to remain consistent. If they don't, a team trying to determine their historic cloud spend won't be able to determine the difference between their cost changes and the allocation strategy that is assigning more or less of the overall cloud spend to them.

A team-by-team breakdown of costs highlights when changes are occurring. Without individual team-level reporting, data can be misleading. For instance, if one team is optimizing their cloud spend while another grows their footprint by a similar size, it can appear as if costs are flat overall. If a team makes changes to the infrastructure and there's no resulting change to the graphs tracking cost metrics, then you're tracking the wrong metrics.

Is Savings the Goal?

As we've mentioned numerous times (because it's very important), savings is not always the goal. Many FinOps practitioners start out focused on ways to reduce their cloud bill, and that's great. But you must always remember that FinOps is about enabling accountability, ownership of spend investment decisions, and ultimately innovation. As you set goals, you should ask yourself: How can I spend the right amount of money to deliver the biggest benefits to the business? What are the benefits I'm seeking? Do they include increased revenue, faster project delivery, cost efficiency in other areas, or even just happier customers? Could spending more here reduce costs elsewhere, such as labor?

An organization may need to move out of a data center within a certain timeframe, and the cost of missing that deadline is much higher than the cost of migrating quickly into the cloud. Or in another case, it may be crucial for a business to deliver a feature ahead of a deadline, which requires using managed services like AWS Aurora instead of running its own MySQL. Aurora costs more than running internally, but it brings with it the benefit of removing time-consuming administration tasks (e.g., hardware provisioning, database setup, patching, and backups), which changes the

TCO (total cost of ownership) comparison and allows the business more time to dedicate to the migrations.

The optimize phase is about measuring your spending against your goals and determining what decisions you can make now and which actions you should schedule for later to clean up.

To encourage a conversation about how your goals should align to savings versus the speed of innovation, we introduce the Iron Triangle.

The Iron Triangle: Good, Fast, Cheap

The Iron Triangle (or "Project Triangle," as some call it), shown in Figure 14-1, is a model of the constraints of project management: speed, cost, and quality. The common formulation is: "Good, fast, cheap—pick any two." It contends that a project manager can trade among constraints in determining the way a project is completed. A change in one constraint necessitates changes in the others to compensate.

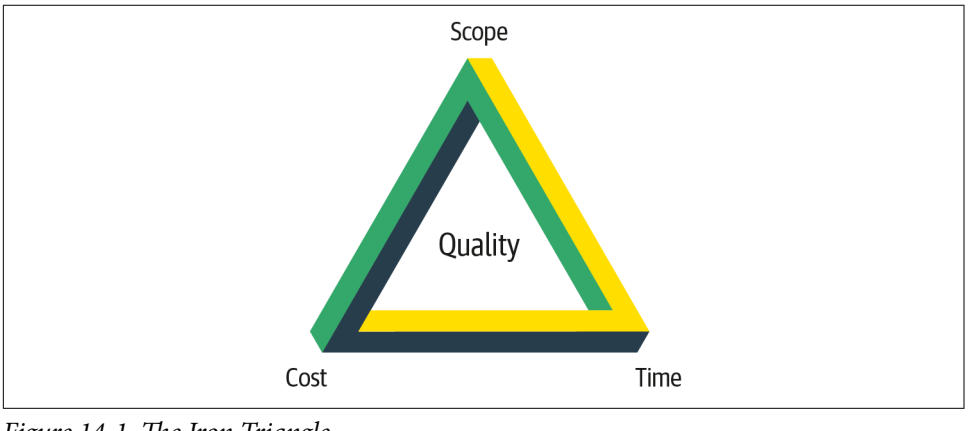


Figure 14-1. The Iron Triangle

For example, a migration project can be completed more quickly if you increase the budget or cut the scope. Similarly, increasing scope may require equivalent increases in budget and schedule. Cutting budget without adjusting schedule or scope will lead to lower quality.

The Iron Triangle is a perfect metaphor for visualizing the process that occurs around cloud decision making—there's often a trade-off among good, fast, and cheap. The more a team aligns with one end of the triangle, the less they focus on the other areas. But this isn't a bad thing. You want teams to make intentional choices toward one end of an axis to support their business goals.

- *Good* measures the quality of your services. Giving more resources to a service running in the cloud can result in improved availability, disaster readiness, performance, and scalability.
- *Fast* measures how quickly your teams move. That speed can be very valuable when you're getting the latest updates or features out to customers, quickly migrating services into the cloud, or cutting time to market for new products.
- *Cheap* measures the amount of cloud spend. A focus on cheap (due to COGS restraints or budget caps) will result in the lowest cloud costs, but with it comes impact to quality and speed, or good and fast.

The way this plays out in the real world was reflected in a recent project at Atlassian. One team oversized resources to gather extra performance data, and, as expected, that resulted in improved migration speed. Had the team stopped to resize resources for cost savings, the overall project would have been delayed. Delivery speed was more important than the amount of potential savings on the table. Once the project was delivered, the focus shifted away from speed, and the team actively began reviewing the deployment to discover possible savings.

We must emphasize that the Iron Triangle isn't intended to be used only at the organization level. It must extend to the team level as well. It's common, and useful, for a cost-aware organization to have a single team and/or service operating with a speed focus. While overall the organization is trying to maintain or increase savings in the cloud, the increase in savings can be reinvested in the team moving more quickly to develop the next competitive advantage for the organization.

Hitting Goals with OKRs

OKRs, or *objectives and key results*, are a framework for defining and tracking objectives and their outcomes. For each OKR, there's an objective to be achieved, along with a set of metrics called key results that will measure the achievement of that objective. OKRs typically have a shelf life of a quarter.

During a FinOps Foundation presentation, Joe Daly, formerly Director of Cloud Services at Nationwide, said, "We call our shots with OKRs and focus on results to provide clarity, accountability, and measurable outcomes."

Daly also made clear that the most important thing with OKRs is to focus on results. When an enterprise is moving to the cloud, there can be a massive amount of disruption as teams try to quickly assimilate a mountain of new knowledge. And that can be very intimidating for teams who have done things another way for years.

OKR Focus Area #1: Credibility

When Daly's FinOps team at Nationwide was relatively new, he offered this advice: "Credibility is probably the most important area to focus on when you're starting up a FinOps practice. Credibility equates to trust, and if you don't have that trust, you're constantly trying to prop up the services you provide."

Daly supports his credibility objective by providing transparency from the end user all the way down to the code. A key part of that is regular spend updates (daily, weekly, monthly) at whatever granularity each stakeholder needs. These updates must be simple and easy to understand. The numbers must also tie to what their accountants will report to them at the end of the month.

OKR Focus Area #2: Maintainable

All too often, new FinOps teams approach their work in unmaintainable ways, such as not enforcing automated tagging. As Daly says, "Meaningful data needs to be managed by the people to whom it's meaningful. What we've done is create a tag repository that's maintained by the application product managers and business-facing folks, so that you can tie all the application data and business data to the resources without depending on engineers, to whom the data is not as meaningful."

Two examples of key results his team has set in this area are being able to tie application name, application owner, and business group to each resource and automating routine, time-consuming tasks like chargeback.

OKR Focus Area #3: Control

The goal here is to focus on establishing control while also enabling speed. In Daly's words, "We push accountability for usage control to the application/product teams." They've accomplished this by establishing a direct chargeback model, sharing knowledge, and encouraging user adoption, while being sure to implement policies to protect against autoscale nightmares.

Some examples of key results his team has set in this area are to double their tag compliance rate, to establish chargeback for cloud and container usage, and to automate compliance policies.

Teams have different motivators that will drive spend and savings. Engineers quite rightly set goals around more performance, higher availability, and faster deployments of new features. Finance teams focus appropriately on spending, savings, and efficiency of spend. And business leaders predictably remain focused on overall business performance. When setting goals for FinOps, you can't just leave these teams to individually set their goals. Not only will it not achieve the correct outcome for the business, but it will also drive up friction between teams.

When you have engineering and finance talking together you usually get a solution that works for both departments, as opposed to engineers just getting in the room and coming up with a solution that only works for the engineers. Likewise, finance comes in developing policies that make life harder for the engineers. Getting people to work together to build better-focused OKRs will drive better results for everyone.

—Joe Daly, formerly Director of Cloud Services at Nationwide

Business leaders need to decide how much they are willing to spend, and when they should forgo savings (and potentially even incur waste) in the interest of speed. But whether it's about moving IT out of the data center as soon as possible or getting a new service out to customers, there should always be tracked spending expectations. Speed at all costs works only until spend grows much higher than is acceptable.

Instead of waiting for cloud costs to breach uncomfortable thresholds and then struggling to respond, setting early expectations and tracking cloud spend as projects progress will help avoid bill shock. Engineers are then given more freedom within agreed cost bounds. FinOps teams can help identify the easy wins to keep budgets on track, while finance and business leaders are able to reassess budgets and targets.

In FinOps Certified Practitioner training sessions, the topic of the FinOps team's role in the optimize phase is about helping to make the important decisions about cost that keep spending in bounds (no matter how fast you're trying to go) and then being able to identify places to optimize later when ideal cost paths aren't followed (due to speed OR due to sloppiness, lack of skill, desire for higher quality, etc.). So we find the low-hanging fruit to clean up, but we also inject an important set of cost checks early in the process for governance as well.

—Rob Martin, Director of Learning at the FinOps Foundation

The entire business needs to find this happy medium, spending enough to enable technology teams while keeping spending within acceptable bounds. Ultimately, you're building toward a FinOps nirvana, where unit economics will enable you to determine the amount of business value you gain from cloud spend.

Discussing goals with FinOps practitioners at other organizations can help you with building out your own goals. Hearing the goals that are important to them, and why, can assist you in ensuring you are setting the right goals and aligning with the general FinOps community.

Stories from the Cloud—FinOps Community

Here are some sample goals shared by a member at one of the FinOps Foundation's mini-summit calls:

> We have a $3m cost-avoidance goal this year. We plan to achieve that by ramping up our reservation coverage to 80%. We break down our reservation coverage by budget tranches. As we continue to buy reserved instances, we want to ensure a high

reservation utilization. We have a goal of 95% reservation utilization, and track that by budget tranches as well.

We have a KPI for tagging, as it's critical to our optimization efforts. The application owner tag is critical to our efforts around rightsizing. We also track rightsizing opt-out. We track who has opted in and out of the recommendations we provide to them. Our goal is to get to only 25% of rightsizing recommendations being opted out of.

Also, we're tracking our cloud provider budgets across two things: (1) our commitment to the provider and (2) our budget progress within each budget tranche. Finally, we track the total opportunities to save via rightsizing and reservation coverage, and how that number changes over time. We have a target for the year for how low we want to get the total savings opportunities.

Goals as Target Lines

Goals aren't just a single value—they're a trend over time. If you set a goal as a single value such as *x* dollars per month, you have to wait until the end of the month to see how you are tracking. By breaking this goal into daily values, you can draw target lines on your cloud spend graphs that enable near-real-time analysis. Target lines are critical in metric-driven cost optimization, which we will cover later in Chapter 22.

Metrics should always have target lines to provide more context to the data. Where you set the target line will be based on your organization's cloud journey, how aggressive you are in maintaining spend, and the value you see in spending more to enable innovation.

Organizations currently focusing on the "fast" corner of the Iron Triangle might set their targets pretty high. Breaching the targets will only be informational and will result in raising the forecast. On the other hand, a cautious organization (focusing on the "cost" corner of the Iron Triangle) will be setting targets fairly close to its existing spend trajectory, and breaches in spend will be followed up by quick actions to get back on track.

If you graph your spend over time as in Figure 14-2, you can determine a few basic things:

- You're currently spending between $400,000 and $530,000 per day.
- Overall, your cloud spend is increasing.
- Last month's spend increased at a steeper rate than previous months.

But these basic facts about the cloud spend are only data points. You can't determine any performance or business expectations from the graph.

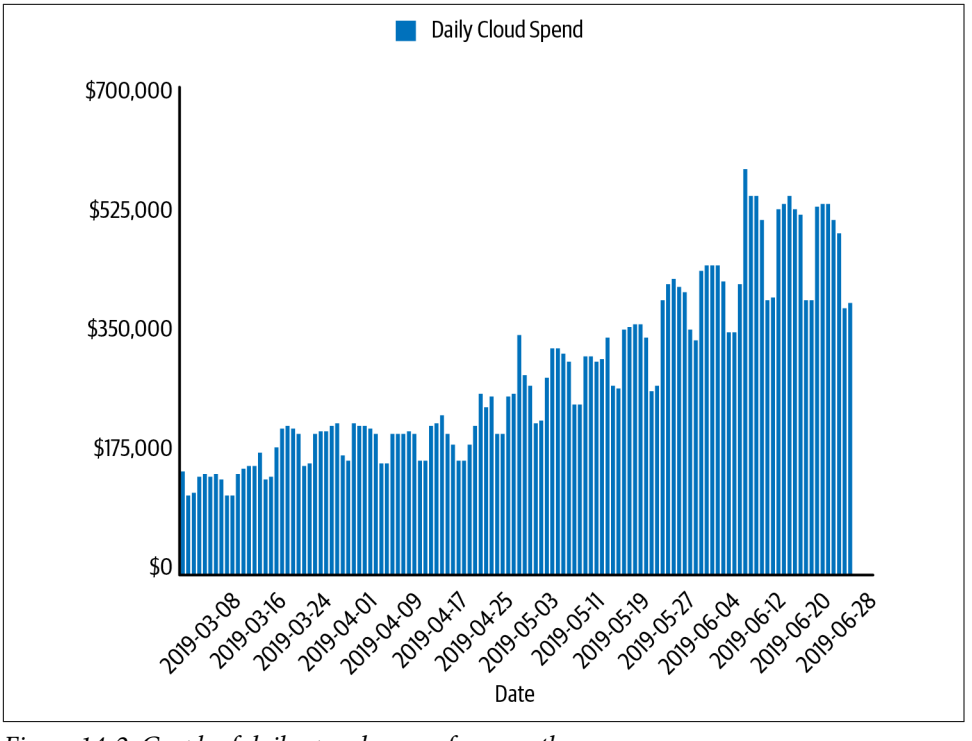


Figure 14-2. Graph of daily spend over a few months

However, if you add a target line, as shown in Figure 14-3, you can determine a lot more:

- Your spend has been under target historically.
- Last month you overspent against your target.
- You haven't revised your target over the previous months.
- You will be over target again this month if your current trend doesn't change.

Adding the target line allows you to get more context about the graphed data points. A target line does not always need to be linear; consider a target that has a quick ramp up and then a plateau or delayed growth then rapid rise. Where possible, you should always try to include a target line within your charts to understand the impacts of the data points on the organization. This plays into the language of FinOps, as discussed in Chapter 4.

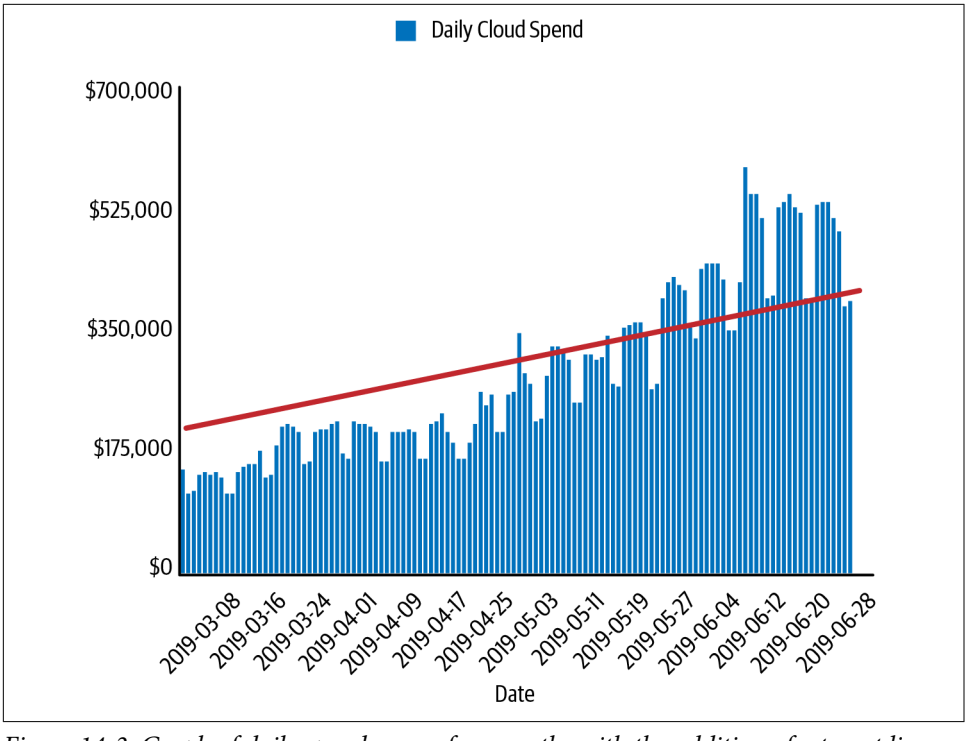


Figure 14-3. Graph of daily spend over a few months with the addition of a target line

Budget Variances

Generally, if your metrics vary significantly from your targets, there's an issue. To maintain budgets, teams should take breaches of targets seriously and react to changes in metric performance as early as possible.

While it might not appear to be important if metrics are way under targets, you should always strive to set your targets at the level expected. This will build confidence in the numbers being tracked and ensure that all details are being built into the target-setting phase.

Sometimes teams need to make choices, and those choices will affect their budgets. Consider a case in which one team is behind on a project. Their plan was to deploy smaller cloud resources to keep costs down, but to get the project back on track, they decided to deploy larger resources and complete things faster. This will result in a short-term breach of targets, but it will have a business benefit of getting the project delivered on time. This is what we classify as an *expected anomaly*. A decision is made ahead of time to exceed the planned budgets to achieve a desired business outcome.

An *unexpected anomaly* occurs when teams deploy resources with the expectation that they will maintain budget, and then find they are trending over budget. An unexpected anomaly typically means something will need to change: either what was deployed, or the targets set for the team.

Say, for example, that new features in your products could drive up customer interaction with your services, which would in turn increase costs. If that increase in cost also increases your earnings, then this would be a good anomaly to see in your cloud spend. (We'll discuss unit economics, a process that allows you to track earnings against your costs, in Chapter 26.)

You must also be able to track anomalies that might not directly result in a change in cloud spend. If one of your teams starts using a new cloud service offering, replacing the usual one, you can learn of this through anomaly reports that show your cost by cloud service offering. Anomalies in this report can be very significant for companies that require sign-off—for security or compliance reasons—before using new services.

Tracking your cost optimizations allows you to identify unexpected changes in optimization performance, which could indicate that part of your FinOps strategy is not being followed correctly. Anomalies in the optimization reports allow the FinOps team to react early in order to maintain the savings that an organization has come to expect. Using automated, machine learning–based anomaly detection is key to finding those needles in your cloud haystack quickly.

Using Less Versus Paying Less

When you know what you are spending versus what you are expecting, the next obvious question is what to do when you are over forecast. Well, there are two ways to reduce your cloud bill, and to best understand them, let's revisit how the cloud charges you.

We covered how cloud providers charge you in Chapter 5. Remember, the basic formula for cloud spend is *Spend* = *Usage* × *Rate*. To reduce spend, you can either reduce the amount of *usage* you have or use cloud service provider–specific offerings to reduce the *rate* you pay for your resources.

You reduce the usage by either reducing the size of provisioned resources (e.g., smaller virtual machines with less virtual CPU [vCPU] and memory) or turning them off/removing them when not needed. You generally earn a rate reduction by making a commitment to your cloud service provider for a set amount of usage over a period of time. In return, they reduce the rate you pay for those resources.

We'll be covering tactics and strategies for each of these two levers—usage and rate reduction—in Chapters 15 and 16.

Conclusion

The inform phase of the FinOps lifecycle helps you understand where things *are*. As you move into the optimize phase, you set goals about where you *expect to be*. Using these goals enables the business to focus on items that need attention.

To summarize:

- The optimize phase is not always about reducing costs—it's about spending efficiently.
- Goals give context to metrics, allowing people to understand not only where things are, but also where they should be.
- Anomalies detected early can be addressed quickly, avoiding billing surprises.
- You can reduce costs through usage reduction or rate reduction.

When reducing cloud spend, it's important to understand what is actually needed. Reducing spend at the cost of innovation, or at the cost of impacting an important project, should always be avoided. Reducing costs in the cloud is complex, but we'll help by breaking it down over the next few chapters.

CHAPTER 15

Using Less: Usage Optimization

In this chapter we discuss one of the toughest parts of the FinOps lifecycle. Usage reduction can take lots of effort and time to implement culturally. If cost-saving quick wins are your goal, you will find that purchasing commitment-based discounts—which we cover in Chapter 17—is a faster path to cost reductions. Unlike buying reservations, which can bring dramatic cost savings without any engineering action, usage reduction requires engineers to insert optimization work into their sprints. This isn't to say there's no low-hanging fruit—just that it will take more effort to get it done.

Cloud service providers sell you on the idea that you pay for what you use. However, a more accurate statement is that you pay for the resources you provision, whether you use them or not. In this chapter, we're going to look at the resources deployed into your cloud accounts that are either not being used or not being used efficiently.

The Cold Reality of Cloud Consumption

When you start shining a light on usage optimization, don't be surprised if you hear a version of this conversation:

> **FinOps practitioner:** "Hey, it looks like you're not using these resources over here. Do you need them?"
>
> **Team member:** "Oh, those are still there? I forgot about them! I'll delete them now."

At this point, you may, understandably, slap a frustrated hand to your forehead. You'll quickly realize that if you had simply had this conversation sooner, you might have saved months of paying for that resource.

But it's not that simple. When you're operating at scale in the cloud, you can almost guarantee that there'll be some underutilized or forgotten resources. When faced with the seemingly limitless resource choices in cloud, many engineers will tend to go for the larger resource option. Making that choice can keep them from getting paged in the middle of the night.

However, it can be difficult to know which resources need reviewing. As we've mentioned, it requires that teams have the right metrics to understand their application's requirements. Teams must also understand any business reasons for why resources are sized and operated the way they are. FinOps is so effective across organizations because it requires that all groups around an organization follow a consistent workflow and work together in putting the recommendations into action.

However, when you do engage usage reduction workflows, you can realize savings where you previously have been wasting money on oversized resources, and you can prevent waste from building up inside your cloud usage into the future. We'll look at what constitutes waste, ways to detect the resources needing investigation, methods to reduce usage, how to get your teams to implement changes, and the correct means to track your progress.

Usage reduction at scale requires a cultural shift. It's not a one-time fix but a continuous cycle of picking properly sized resources and eliminating overprovisioning. Usage reduction requires engineers to think of cost like they think of memory or bandwidth: as another deployment KPI. And they'll see very soon how thinking about cost as an efficiency metric is worth the effort.

Halving the size of a resource will generally save 50% of the cost—or in the case of an unused resource such as an unattached block storage volume or idle instance, 100% of the cost. One of the benefits of the variable, on-demand nature of the cloud is that you are free to resize any resource—or to stop using it entirely—whenever you desire. Further savings are possible on correctly sized resources that you use consistently with rate reduction options, which we cover in Chapter 17.

Initially, usage reduction tends to be retroactive. Teams go through and clean up their infrastructure to ensure a fully utilized set of resources. In more mature practices, engineers are proactively considering cost in architecture reviews before deployment and actively considering cost when choosing cloud resources initially. Later phases bring the rearchitecting of applications to use more cloud native offerings.

The goal isn't to make the perfect decision. It's to make the best decision possible given the context and data, and then reevaluate often. A well-sized resource won't stay that way forever, as code and load evolve over time.

Stories from the Cloud—Mike

At Atlassian, Mike has seen a big shift since the first edition of the book:

> Four years ago, FinOps was seen as demanding time from engineers and generally just getting in their way. The practice has matured to be more integrated with engineering itself as it's moved to being a core part of delivering software in the cloud. The FinOps Foundation has also seen membership from engineering roles grow considerably in the last few years. Before FinOps was an established term, you'd go to your engineers and say, "We're trying to do FinOps or cost optimization," and they didn't really know what that meant. They've now generally heard the terms and have an understanding of what you're trying to achieve. The effective FinOps teams arrive as partners, not outsiders. The FinOps teams have gotten better about softening the tone and look for collaboration with engineers versus just pointing the finger at wastage.
>
> In our FinOps training at Atlassian, we do an exercise where we look at two different models. First, we go into conversations with a demanding, blame-focused approach (i.e., "You're doing something wrong"). Alternatively, we turn up with blameless data and ask to discuss the reasons why the data is indicating underutilization. The exercise drives home that it's not just about finding waste, it's about working and collaborating, seeking first to understand.

Where Does Waste Come From?

What we call *waste* is any paid usage or portion of paid usage that could have been avoided by resizing or reducing the number of your resources to better fit the needs of your application.

When cloud resources are first deployed, often their capacity requirements aren't fully known or well understood. That's true for almost everyone in those early days of deployment.

This uncertainty leads to the teams overprovisioning capacity to ensure there won't be any performance issues. Overprovisioning resources when initially deploying isn't a bad thing. The beauty of cloud is that it gives you the ability to deploy quickly and then make adjustments along the way. What you want to avoid is deploying resources and not monitoring them for overprovisioning. That's when you find yourself paying more than you should for resources over a long period.

Wastage grows as teams neglect to maintain their resources based on utilization metrics. It is essential to continuously monitor your resources for oversizing. That's because a service that's using its resources correctly today may become overallocated tomorrow after the deployment of more efficient service code. Even changes in customer behavior can result in reduced capacity requirements.

Usage Reduction by Removing/Moving

Research shows that there are two types of cloud teams: those that forget about some of their resources, and those that fib about forgetting about some of their resources. In other words, it is very common for resources to be forgotten about and left in cloud accounts. Don't assume bad intent or carelessness when you find the same in your company. The job of engineering has many competing priorities, and automated tooling to detect waste is not always prioritized by leadership. In other words, it happens to the best of us.

Find waste by looking for areas with very stable costs and that use dated resources. "Old and unchanging" is not how the cloud should operate. If it's been sitting there untouched for a long time, it likely can be optimized.

Simply telling teams not to forget about resources doesn't cover all the bases. It's possible a resource was created by automation that wasn't configured to remove it when instructed to do so. Or the resource was intentionally left there by the team until an unspecified later date, but no follow-up action was taken as priorities shifted. Additionally, some resources can be configured to remain when their parent resource gets deleted, like volumes attached to a server, and when deleting some resources, the cloud service provider can automatically create a snapshot of the data that sticks around whether you need it or not.

A lost or forgotten resource is the easiest type for teams to handle. Usage reduction in such cases can be as simple as deleting the resource, saving 100% of the cost of that resource.

Another conversation we often hear is around the need to keep data stored inside a resource. Usually, teams have compliance reasons to retain data for very long periods.

Even if you need to retain an unused volume for this purpose, there are ways to reduce what you are paying. For example, cloud service providers offer multiple storage tiers, and paying for data in higher-price storage offerings makes no sense when you can move it to a lower-price "cold storage" resource like Azure Archive or AWS Glacier storage.

Consider creating a data retention policy that makes it clear to teams what data they need to keep and for how long. Once a retention policy is in place, teams can automatically move data to different storage offerings and delete it when appropriate.

Usage Reduction by Resizing (Rightsizing)

To resize a resource, you need visibility into how much of the resource is utilized. That visibility must include CPU usage, memory utilized, network throughput, and disk utilization.

When rightsizing, you aren't just optimizing for cost savings. You also need to make sure that you don't impact the services that teams are operating. The primary goal of the teams when managing services is to ensure that the services themselves do not run out of required operating capacity. As a result, it's not uncommon for them to resist the idea of resizing resources, especially those supporting production applications.

If the teams have to stop working on what they are doing to investigate resizing their resources, there can be real impacts to delivery schedules. You should consider these impacts when deciding whether teams should focus on resizing resources to reduce costs. Often there are a handful of high-cost resources and a long tail of small resources that would save very little. We recommend a cutoff point (a minimum savings amount) under which rightsizing recommendations can be ignored. This threshold should be reviewed often to be sure it remains sensible. The goal is to make sure that time spent on rightsizing results in material savings for the business.

To prevent concerns about impact to resources or automation while implementing resizing changes, it's important to understand the role the FinOps team plays in rightsizing. FinOps is about collaboration between teams and should not be performed independently by the FinOps practitioner. This is where having a conversation is crucial, because it's usually impossible to infer the method teams used in determining the size of their resources just by looking at metrics alone.

The central FinOps team is there to operate the automated analytics into resource usage, provide reports on what resources appear to be underutilized, and programmatically provide alternative, more efficient configurations. Teams are then given the recommendations for investigating the resources to identify any reasons that would prevent resizing them to realize the potential cost savings. It's imperative that engineering organizations understand that the FinOps team is not intending to introduce performance risk, but focusing on safely reducing capacity where it offers no business benefit.

It's essential to supply multiple recommendations that would fit the existing workload on the resource without affecting workload performance. Multiple recommendations give teams the opportunity to balance the risk involved in reducing the resource size against the potential savings. Native, open source, or third-party FinOps tooling can do the heavy lifting of providing these recommendations, and engineering teams can investigate further using application performance monitoring tools.

Rick Ochs, Principal Product Manager at AWS, shares that it is important to ensure that the rightsizing recommendations are high quality, helping engineering teams avoid hours of validation work to uncover potential risks, such as account quotas, attachment limits, or CPU speed differences. Offering rightsizing recommendations that are directionally accurate but simplistic can often lead to engineering teams spending more time on rejecting recommendations than the benefit of rightsizing can provide.

By using high-quality rightsizing recommendations—those that use more than just CPU average, but include I/O, throughput, memory, and don't use peak or average metric—you can lower the amount of effort required and increase the number of recommendations that turn into material cost savings.

Every rightsizing recommendation is an opportunity to have a discussion with someone. You have to fully understand the overall effort to rightsize existing resources and ensure that new apps are rightsized from the beginning.

Stories from the Cloud—Benjamin Coles

Benjamin Coles, a Senior Software Dev Engineer in Cupertino, offers this advice to his engineering partners:

> For migrations, the lift-and-shift paradigm is a quick and dirty method that allows us to cram as much as possible into the cloud without thinking of requirements. It essentially becomes deferred tech debt; this is the process in which we kick the can down the road without seeing if something can be tuned.
>
> This is not the ideal long-term solution; if we had the time, we'd refactor the code—we'd do it right. In the meantime, what can we do as small wins to make the move more bearable? If you have monitoring turned on for your source instance (bare-metal or virtual machine [VM]), take a look at CPU/memory and see how much your application is using. Instead of using the instance specification, you could build an EC2 that matches closer to what your performance requirements are on your instance.
>
> Why would this matter? In the traditional sense, when you use an on-prem instance, you pay once and never think about it again. In this new cloud world, they've itemized every subsystem down to the cost of CPU and memory. In the cloud, there is a death by a thousand cuts as you are charged for storage and network I/O and every other service you sign up for. Even if you were to focus on the instance costs, ignoring all the related subcosts, there is a lot of money to be potentially saved.
>
> For example, If you had 100 systems with the 16 core by 128 GB of RAM—that's equivalent to a c6g.4xlarge, which prices out to $0.544 an hour, or about $476K a year. Reducing your instance size by half would yield a c6g.2xlarge at $0.272, or $238K a year run rate. What works with this model is that you can garner favor with

your finance team as you've reduced operating expenses and even better is if you could do that same thing again when you trigger a redeploy.

Benjamin's story especially makes sense when you consider that during the time we built things in the data center, we intentionally made infrastructure oversized to allow the workloads on them to grow over the five to seven years they'd be in service. It was considered a standard best practice to oversize your physical hardware purchases to avoid the pain of having to repurchase hardware if your applications cause the system to run out of capacity only halfway through the hardware's life span. This has caused a common outcome of very low single-digit average hardware utilization in physical data centers. If we lift and shift by matching exactly the CPU and memory that was in the data center, we are carrying over that oversizing without even knowing it. An alternative approach if you can't look at utilization right away is to go a little smaller in the cloud by default, and address issues of performance as they come up. This approach can help lift and shift cost efficiency if engineering can easily identify performance issues but not yet see the cost.

Common Rightsizing Mistakes

Rightsizing is rarely as straightforward as it seems when first considered. There are a plethora of details to consider, and many common mistakes are repeated by those new to cloud optimization.

Relying on Recommendations That Use Only Averages or Peaks

When looking at the existing workload on a resource (like a server instance), it's crucial to ensure that you're not just looking at average and peak values of the metrics. If you use average or peaks, you can be misled.

It's not a simple matter to take into account shifts and spikes in utilization and then apply a statistical model to enumerate the best-matched resource size. Most tools that we've seen use some kind of rule of thumb based only on utilization averages over a time series, and then recommend the tightest-fitting resource type. Other tools will recommend upsizes for any peak they see, no matter the reason.

For example, when you compare the two usage graphs in Figure 15-1, both have an average CPU statistic of 20% utilized over the hour. However, using the average CPU usage metric, you can't tell whether the CPU was at 20% the whole time (as shown in the right graph) or up near max CPU for one-fifth of the time and near minimum for the remaining four-fifths (as shown in the left graph). Resizing these instances to a server instance with half the CPU capacity would impact the performance of the left instance during its peak CPU usage. It doesn't matter if you look per day, per

hour, or even per minute. Averages don't tell the full story of what is occurring on the instance.

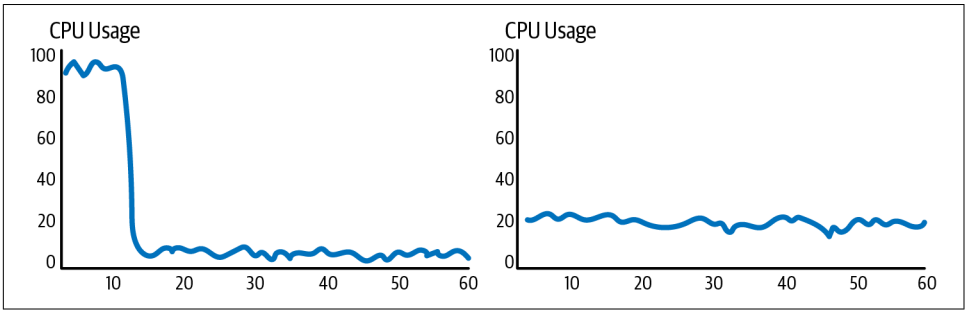


Figure 15-1. Graphs showing two CPU workloads: one has a short 90%+ peak and then remains at <10% for the rest of the hour, while the other is consistently at 20%

Using only a utilization average can lead to two outcomes. The most common scenario is that the engineer realizes the recommendation is completely bogus, and the cost-saving initiative ends. The team realizes that they've wasted a bunch of time, and their deployment is still costing more than it should.

The other scenario is far worse. A less experienced engineer may go ahead and take the recommended action. During quiet times, or even average usage times, the new resource configuration will likely hold up fine. But when a regular spike or shift occurs, things begin to unravel. Demands on the resource push it beyond what it's capable of. Performance starts to degrade, and it's even possible to see some sort of outage. Any potential cost saving becomes vastly outweighed by the risk to the business.

> *Sizing all of your workloads to max—or peak—can also be a mistake. In a typical month of enterprise VM usage, various activities will occur on these workloads: patching, maintenance, deployments, backups, and the like. While any of these activities can spike CPU usage, that does not mean you should immediately upsize, or trust a recommendation that always sizes your instance to cover a peak usage spike. Even normal OS reboots will drive a CPU utilization to 100%. That would be akin to throwing out your laptop and buying a faster one every time you boot up, because it hits 100% CPU utilization. Instead, using a sizing method that will remove a small number of peaks from the rightsizing is preferred, where transient spikes due to patching, maintenance, or other behaviors do not drive upsizes, but only utilization sustained for more than a reasonable 1% or 5% of the time over a week. Ideally, use rightsizing recommendations that account for this by using percentile calculations.*
>
> *Imagine if you were using a rightsizing product that only keyed on peak usage, and then a major security vulnerability was discovered that required a large patch to be installed across your entire company's fleet of instances. Your rightsizing recommendations would erroneously tell you to upsize everything.*
>
> —Rick Ochs, Principal Product Manager at AWS

Failing to Rightsize Beyond Compute

Everyone loves to first focus energy on compute savings, but rightsizing needs to extend beyond compute so that you solve for the bigger cloud cost picture. You see utilization issues across the board, but two particular spots worth mentioning are with database servers such as RDS (Relational Database Service), managed SQL, Azure managed disk, Cloud SQL, and storage such as EBS (Elastic Block Store) and Google Cloud Persistent Disk. If you aren't looking at noncompute services such as database and storage, you're leaving a whole bunch of potential savings on the table.

Not Addressing Your Resource "Shape"

Rightsizing decisions shouldn't be stuck within one compute family. Each cloud provider has a number of specialized compute families, and you should choose one that matches the resourcing shape of your workload. Within AWS and Azure, these families are offered in fixed shapes with a certain number of CPU cores, a certain amount of memory, and potentially other features like fast networking. For example, your workload may require four cores and 3 GB of memory, and might have been moved to the cloud running on an r6i.xlarge in AWS. This has the right number of cores, but dramatically too much memory. Moving instead to a c6i.xlarge not only reduces the cost by over half but also gets your shape correct by keeping compute consistent while at the same time reducing your memory. The way Google Cloud compute is purchased allows you to select the CPU and memory mix instead of picking from a large list of predefined instance configurations. You may also find that certain instance families have faster CPUs, which can sometimes allow you to lower the core count, or, alternatively, slower CPUs, which are offered at a lower price. By being thoughtful about the instance families you can pick from, you can achieve additional savings beyond simply picking the correct count of CPU and memory.

Not Simulating Performance Before Rightsizing

Your teams might be concerned with clipping (impacting performance when reducing a server's resources), and that can drive suboptimal decisions. But with the means to get a forecast of how a recommendation affects their infrastructure, they can make better choices to optimize their cloud. Before making a rightsizing move, you must be sure to visualize the impact of each option and consider multiple recommendations across compute families. That way, you can assess the probability that there could be clipping (see Figure 15-2) and take that risk into account when compared to the savings you'll realize. Without this step, you risk being too conservative (limited savings) or too aggressive (performance hit/outage).

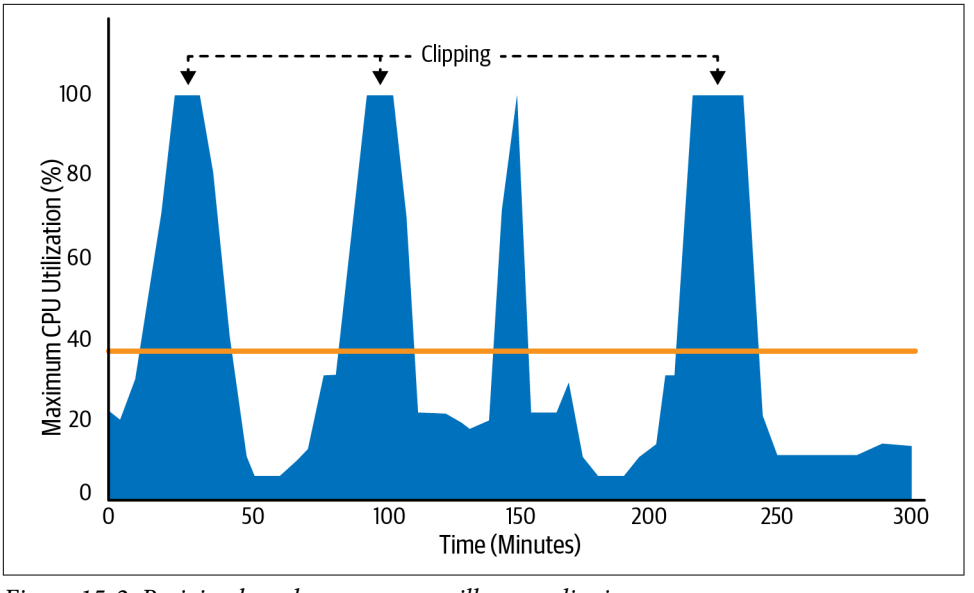


Figure 15-2. Resizing based on averages will cause clipping

Hesitating Due to Reserved Instance Uncertainty

A common refrain from teams is, "I don't want to rightsize because it might affect my commitment coverage and cause waste." There was a time when this was a valid worry and caution was required. But not anymore. All three cloud providers now have a number of options that give you far more flexibility and allow you to rightsize with confidence. When using reservations that have flexibility, you're able to adjust your reservations if they're impacted by your usage reduction efforts. We'll cover these flexibility concepts in Chapters 16 and 17.

Going Beyond Compute: Tips to Control Cloud Costs

While compute resources make up a supermajority of cloud spending on many cloud bills, there are many other usage optimizations that are important to consider to cover other services that commonly can be optimized. Here we'll look at some examples of other cloud service focus areas.

Block Storage

Closely related to cloud compute instances are the block storage devices attached to them. Looking at the big three cloud service providers, block storage services include AWS's Elastic Block Store (EBS), Azure managed disk, and Google Cloud's Persistent Disk.

Get rid of orphaned volumes

The major feature of block storage is that the volume persists after the compute instances stop. This is good for data retention, but bad if you're paying for storage that's no longer needed. Unused volumes are called *unattached* or *orphaned* volumes, and they're marked as *available* if you check their status. These volumes can't take any traffic without an attached compute instance, so they're useless as is. A great first step to saving money is to get rid of them.

One option is just to terminate an unattached volume, but you should do so only if you're confident that the data is never going to be needed. To be sure, you should see when the volume was last attached. If it was months ago, there's a good chance you no longer need the volume. This is especially true for nonproduction environments. Some cloud providers allow you to leave the block storage available when a spot compute instance is removed, so before taking action, you should be sure you don't have any data retention needs for this data.

There's a great story about how the deletion of a nine-year-old unattached volume created a production issue on Episode 13 of the FinOpsPod podcast entitled "FinOops: Lessons Learned the Hard Way." Be sure to check it out on Spotify or Apple Music.

A more cautious approach is to first take a snapshot of the volume and then terminate it. Snapshots are always cheaper than the original volume. They discard blank space, compress the data, and are stored in cheaper storage tiers like AWS's S3 or Google Cloud's Cloud Storage. If you do need to restore the volume, you can do it from the snapshot.

Focus on zero throughput or zero IOPS

Once you've gotten rid of unattached volumes, you look for attached volumes that are doing nothing. These often show up when the associated instances have been turned off and the volumes were forgotten. To discover these volumes, you look at the volume network throughput and IOPS (input/output operations per second). If there haven't been any throughput or disk operations in the last 10 days, the volume probably isn't being used.

Make managing your block storage costs a priority

With block storage volumes, you pay for two key attributes: storage and performance. Storage is charged per gigabyte stored, with a rate based on the location and volume type. For performance, the better it is, the more expensive it is, whether in terms of IOPS or throughput. Amazingly, volumes are often ignored when optimizing cloud expenditure.

Reduce the number of higher IOPS volumes

Volumes with higher IOPS guarantees (e.g., AWS provisioned IOPS volumes or Azure premium disk) aren't cheap, and it's relatively easy to change them. Using historic metrics to locate underutilized disks and having engineers change them where possible can greatly reduce disk costs.

Take advantage of elastic volumes

When you use AWS EBS, a volume can increase in size, adjust performance, or change volume type while the volume is in use. This can be done while an application is actively driving I/O to your volume. There's no need for a maintenance window or downtime. There's a big cost savings benefit, because you no longer have to overprovision your storage. Often engineering organizations will be far more open to taking recommendations that require no downtime, as they don't have to deal with change tickets, downtime windows, or the risk of having to take additional downtime if they want to revert the size change.

Object Storage

Often referred to as *unlimited storage*, object storage services are highly likely to be the location with the majority of your data. Object storage services include AWS's S3, Google Cloud's Cloud Storage buckets, and Azure's Blob Storage.

Implement data retention policies

The ease of storing data into object storage services and the fact that these data stores are theoretically limitless makes it easy for engineers to store data forever. In the data center, there was available disk space as a constraint and data retention policies had to be implemented to avoid unbounded storage growth. Moving into the cloud removes the challenges of making storage capacity available, but it should not remove the need for data retention policies.

Choose a storage tier/class that matches the data

By default, data sent to object storage services is often stored in the standard/hot storage class. Standard storage is usually the most expensive storage class but has the most availability and durability. Classifying your data and selecting the right storage class for the data type can save you significant amounts on object storage. Some cloud service providers provide automated classification and data class selection, such as AWS's Intelligent-Tiering.

Networking

Networking costs can often be overlooked, possibly due to the fact that network traffic is not a named resource like a cloud compute instance or an object storage bucket. There are a few ways to optimize networking costs, but many will need strong collaboration with the team that manages the network within your organization.

Clean up unused IP addresses

It's possible to be assigned fixed IP addresses from your cloud service provider; when these addresses are unassociated with a running compute instance, they often have an hourly charge. Unless there are business reasons to keep unused IP addresses, removing them can reduce some networking spend.

Optimize network routes

Some cloud service providers offer networking constructs (like AWS VPC endpoints) that enable access to their service APIs without using a public IP address or routing traffic via network address translation (NAT) services. Using these network constructs can reduce the cost of transferring data between your applications and the cloud services in use.

Usage Reduction by Redesigning

The most complex method of usage reduction is to redesign the services themselves. Having engineering teams change the way software is deployed, rewrite applications, or even change the software altogether can help you take advantage of cloud native offerings.

Scaling

Changing the size of a resource isn't always the whole answer—the resources might be utilized perfectly well during business hours but not outside of them. Of course, if you resize the instance to be smaller, during business hours it will be overutilized and probably cause performance issues.

Teams might be able to architect production services to be more cloud native, allowing them to take advantage of the elasticity of the cloud. The most common approach is to have services scale out horizontally (i.e., provision more resources) during busy periods and then scale back in off-peak hours. The service itself needs to support this dynamic scaling, which may require redesigning the application code. Or scaling may not be possible at all, especially with proprietary software.

Modern methods of building more service-based, loosely coupled, modular, and stateless architected applications in place of monolithic applications are gaining traction due to their support in cloud and in container environments, so the ability to scale will appear in more applications going forward.

Scheduled Operations

Teams often leave development and testing environments running while they sleep. Consider a geographically centralized development team that's using their resources for 40–50 hours a week and not using them at all for the remaining 118+ hours. If they're able to turn off development resources for around 70% of the week, they can create massive savings for their organization. Depending on where a distributed team is located, there's almost always 24 hours when everyone should be on a weekend. Incentivizing your engineering teams to turn off resources during off-hours by allowing them to use the savings on new projects or engineering efforts can help drive the cultural accountability of cloud spend.

Providing teams with automated ways to turn off resources when they're not needed is the best method of ensuring that resources are stopped out of hours. We'll cover automation later, in our discussion of the operate phase of the FinOps lifecycle.

Effects on Reserved Instances

We're often asked, "What about all of my reservations?" If there are committed reservations such as Savings Plans (SPs), Committed Use Discounts (CUDs), or Reserved Instances (RIs), there's always a concern about changes to your usage causing your reservations to become underutilized. Generally, you avoid this problem by performing usage optimizations before committing to rate optimizations like RIs or CUDs.

It can take some time to implement usage reductions, usually much longer than was initially expected. Almost every day we hear someone say, "I'll make commitments after I clean up my infrastructure." We've found that 9 times out of 10, cleaning up takes longer than expected—usually months—and during that time they end up paying for oversized resources at the higher on-demand rate.

> *Rightsizing is a very intimate thing and a long process. Engineers sweat over their infrastructure. It's not as easy to say, "Well, it should be on a bigger or smaller instance." There are test cycles and real effort required to make those changes. So, we take the approach that if we can centralize RIs and commitments, make good investments, and target low RI vacancy [waste] numbers, then the rightsizing will catch up eventually.*
>
> —Jason Fuller, HERE Technologies

Priorities tend to change, and fires periodically must be extinguished. You should accept these as facts of life and look to get some immediate coverage of commitment-based discounts, regardless of how much waste you think you have. The strategy we often recommend is to start small with something like 20%–25% coverage, and then to slowly grow it in tandem with cleanup efforts. There will be more on creating an effective commitment strategy in Chapter 18.

Unless you've committed to every bit of your cloud usage, usage optimizations should still be possible. Going forward, you should take into account the amount of usage optimizations available to your organization before committing to rate optimizations. Most cloud service providers allow some ability to exchange or change commitments to match new usage parameters. With some planning, you can avoid being locked into the incorrect commitments and start your usage reduction efforts by modifying rate optimizations as you make changes. By rightsizing your instances across different families, you can often increase your coverage and reduce your instance spend at the same time. You can effectively "Tetris" your workloads in a more optimal way by fitting more instances into the RIs and discounts you've already purchased. It is more cost-effective to rightsize and discount your workloads, thereby potentially leaving some unused discount capacity, than it is to ignore rightsizing for fear of risking wasted discounts.

Benefit Versus Effort

When you're looking at usage reduction recommendations, it's essential to consider the possible savings against the engineering effort and risks to your production services. If the amount of time needed to investigate and implement the changes is more than the savings, it might be best to ignore these recommendations. Ideally, you filter out low-value and/or high-risk recommendations.

One of the FinOps Foundation members has their team look at its cloud spend in terms of engineering hours. They consider how many engineering hours the expected amount of savings would result in. If they can save 1,000 engineering hours by using only 3 engineering hours to capture those savings, they'll do the work. If they'll save only 5 engineering hours, then it's less compelling.

Thinking of savings in terms of engineering hours helps teams to think of the savings they generate as a potential new engineer on their team. The more engineering hours saved, the more likely they will get additional headcount approved.

We don't advise making changes without first investigating the impact of those changes. Sometimes teams perform changes—or worse, set up automation to force resource resizing—without understanding their impacts. This can lead to production issues for services.

Before investing time in performing changes to reduce usage, teams should determine the likelihood of reverting changes. If you're expecting the workload to increase over the coming weeks, and the time it takes to roll out the decrease in size would mean you have to increase it almost immediately afterward, it would not be a good investment of time. Also, you must consider other projects, examining the benefits you would realize by making the changes. If you roll out a new service only days after resizing instances that are now replaced, then once again the time could have been spent elsewhere for better benefit to the business.

While the savings can appear small, especially in contrast to your whole cloud costs, it's important to remember that savings compound over time. By removing an unnecessary resource, you prevent being charged for that resource in every month's bill thereafter.

Serverless Computing

Serverless computing is a model in which the cloud provider runs the server and dynamically manages the allocation of machine resources. Pricing is based on actual activities executed rather than on prepurchased units of capacity.

This removes many of the unused or underutilized issues discussed earlier in this chapter. With serverless, you truly pay only for what you're actively using. Unused resources aren't typically easy to leave lying around. Serverless architectures can usually be ready to process requests very quickly, compared to starting new server instances and deploying your software when needed.

The move to serverless isn't without cost, and it's by no means a panacea to the wastage problem. There was recently a healthy debate within the FinOps Foundation on the best way to forecast and compare the serverless costs for an application versus its current compute-heavy architecture. Several exceptional methods were suggested, and in the end the discussion showed how much the practice is still evolving.

Ultimately, the complexity of building any migration plan to serverless resides in execution and has very little to do with cost savings. A recommendation to move from a large server instance to many concurrent serverless executions is meaningless, since the cost-associated savings in doing so pale in comparison to the engineering effort to get there. For the majority of cases, there's little point in worrying about the forecasting or optimizing of serverless, because the real cost is not the cloud bill but the rearchitecting of applications. Put simply: serverless can be cheap, but refactoring for serverless is not. Thus, serverless is often a better option for greenfield projects rather than existing applications.

However, there's an entirely different lens through which you can evaluate serverless: *total cost of ownership* (TCO), or the cost of the engineering teams that are required to build a solution, and the impact of time to market on the service's success and

profitability. Remember, serverless allows you to delegate a lot of responsibilities to the cloud provider. Duties that DevOps engineers would typically handle (server management, scaling, provisioning, patching, etc.) become the responsibility of AWS, Google Cloud, or Azure, which leaves dev teams free to focus on shipping differentiating features faster.

Too often—even in this book—the focus is on the cost of the infrastructure itself. But the biggest cost in software development is commonly the people. Consider this closely when looking at both sides of the serverless argument. The people cost (e.g., salaries) may cancel out any infrastructure savings when you're considering a move from a monolithic application to a serverless one. Coming back to the benefits versus effort, you should consider the overall cost in redesigning services for serverless against the potential for reduced costs.

But when you're building a new greenfield serverless service, the people cost savings may be well worth it. Remember that serverless can both speed up time to market (by preventing you from rebuilding functionality that is available off the shelf) and dramatically reduce ongoing ops requirements. These benefits allow you to redirect resources to building products instead of maintaining servers.

The entire discussion on serverless—as with all things FinOps—should be grounded in the goal of FinOps: it's not about saving money; it's about making money. On your cloud bill, serverless could actually end up being more expensive for certain applications, or it might save a ton of money for others. The real cost you incur to achieve those savings will be the people cost, but the real benefit you gain from serverless may well be shorter time to market. And when it comes to competitive advantage, opportunity costs outweigh many real costs.

Not All Waste Is Waste

If you start yelling at everyone who has resources that appear to be oversized, eventually your teams start yelling back. (Of course, when doing FinOps, you don't have to yell because there are collaboration, trust, and agreed-upon processes between teams.) Successful FinOps leverages the value of cross-functional teams having conversations with each other. Understanding that there are valid reasons to oversize resources can change the language and tone you use around rightsizing. Many a finance leader has created tensions with their engineering leader by approaching an executive with a list of oversized resources, claiming widespread and rampant waste.

Fixing inefficiencies won't always be the most important thing. The prioritization of addressing waste versus everything else needs to be set by engineering leadership. The people making those decisions should be doing so from a place of solid information and understanding of both the cloud environment and the business priorities. Savings opportunities must be communicated in a manner that allows them to be apples-to-apples compared against competing priorities.

Where systemic waste has been consciously decided upon, aim to have a way of filtering those out (ideally with some time bounding so they get revisited later) so they don't create noise obfuscating the truly addressable opportunities. The process of excluding items from review should also have oversight.

Until you have confirmation from the teams responsible for the resource, you can't be sure there isn't a valid reason for overprovisioning. What's important is that someone spends some time investigating the reasons why the resource is oversized and either makes adjustments to its size or provides context for why the resource is sized as it is. This is why you decentralize usage reduction to the teams responsible for each application. While a centralized FinOps team can help provide recommendations and best practices on rightsizing, the ultimate action needs to fall on the application or service owner.

One valid reason for overprovisioning is hot/warm disaster recovery. A resource is sized to allow production traffic to be moved onto the resource fast, meeting the recovery time objectives of the team's service. Another could apply for services that need extra capacity in the event of a failure. During normal day-to-day operation, the service doesn't need the additional size. Nevertheless, it requires it in the event of failure.

Even teams that are fully optimized can be surprised. Optimization opportunities often appear via external factors that are not in a team's control. A price drop, a performance increase, a service release, a change in architecture, or a similar event might trigger the need to optimize or rightsize. Yet the team has little ability to foresee these events or plan for them. So scorekeeping due to external factors may not be entirely fair.

Again, FinOps is about collaboration between teams. Resizing resources without the input of the teams that manage them may result in issues for the services and also for future optimization efforts.

Unless application health and performance are prioritized over savings, engineering teams will tend to view rightsizing with skepticism. Once performance-focused rightsizing recommendations can be trusted and accepted, you will see their acceptance level begin to rise.

Your usage optimization workflow should allow for these reasons to be tracked. You can use a ticketing system to enable you to define this workflow—and to record the communications and investigations. A ticketing system also allows you to track actions over time and work out what outstanding tasks you have and their current status.

Where there's a business case for oversizing a resource, it's no longer classified as waste. Once the team that owns the resource provides a valid reason for the sizing of a resource, it helps to have a process to mark that investigated resource, removing the savings recommendation from your tracking.

Remember that even just the investigation of savings opportunities that aren't ultimately actioned or actionable can be important opportunities for the FinOps team to work collaboratively with the product and application teams, to learn more about the application environments, and to establish trust.

Maturing Usage Optimization

Using a gradual, incremental improvement strategy is a recurring theme in FinOps, and it applies to usage optimization as well. You shouldn't be trying to solve all wasted usage at once. You should instead identify the usage reduction method that is most likely to save your organization money. And in the early stages, that's usually idle resources.

While doing so, you build out reporting, showing your organization the size of the problem and the potential to save. Then you aim to optimize the top 10 items on the list and review the effectiveness of your engineers' efforts. Having that feedback loop showing the results of efforts put in by your teams enables trust in the process, and it shows the organization that continued effort is a benefit.

Starting small reduces the risks to your organization, because many of the strategies that enable usage reduction come with changing resources and/or software that is used to deliver services to customers. Decreasing the number of in-process changes at any one time will lower the chance that these changes might greatly affect the business.

We suggest starting with the lowest-impact recommendations first, such as cleaning up unused volumes and IP addresses. Test out rightsizing recommendations in non-production environments, and get the broader organization used to the variable

usage benefits that cloud provides via its ability to change resource sizing to match needs. This elasticity provides an enormous benefit to your business, but only if your organization can integrate this behavior. Actioning block storage volume rightsizing recommendations requires no downtime and very low risk, another great entry point into the world of rightsizing. Remember FinOps principle 6, which reminds us to *take advantage of the variable cost model of the cloud.*

Advanced Workflow: Automated Opt-Out Rightsizing

The hardest part of usage optimization is having teams take responsibility and perform the actions needed. Recommendations alone don't make you more efficient—the workflow and practice of implementing the changes required is where savings are realized.

Stories from the Cloud—FinOps Community

Here's a real-world example of how a Fortune 500 biotechnology company implements its advanced optimization workflows. The FinOps team is responsible for leading the complete migration to cloud-based computing (AWS and Azure) and global responsibilities for cloud platforms, FinOps, and on-premises computing services and data centers.

The team meets biweekly to review status on various optimization tasks, discuss roadblocks and how to remediate them, and plan for future optimizations such as future rightsizing, RI purchases, RI conversions, and the like that are scheduled on a shared calendar and later loaded to their schedule tracker for planned optimizations.

Figure 15-3 shows optimization tasks assigned by owners with an associated color coding based on progress.

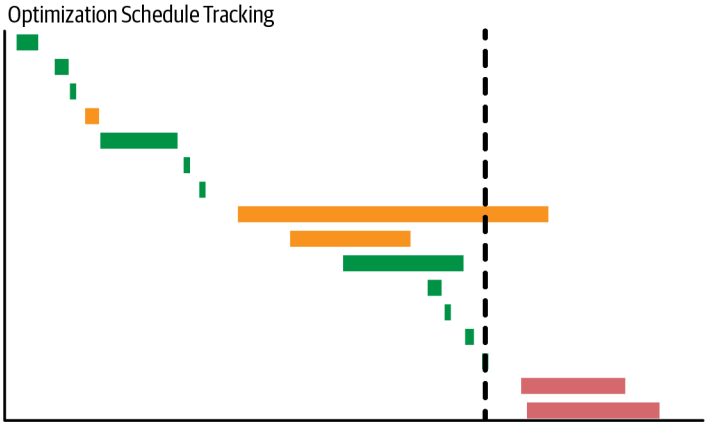


Figure 15-3. Schedule tracking of optimization tasks

The company's rightsizing process is fully automated. At the start, their FinOps team kicks off rightsizing scripts that link to their FinOps platform's API. The recommendations are then queried into the table, and a change request is submitted for change review for all nonproduction instances. If the change is approved, an email is sent to their application owners, notifying them that they have a rightsizing saving opportunity.

An automation workflow (see Figure 15-4) has been developed and broadcasted across their organization, giving everyone confidence in the process. If an application owner opts into rightsizing, it's automatically executed at the date and time specified in the broadcast. Once that rightsizing recommendation has been executed, then they publish the cost avoidance associated with the rightsizing. If teams opt out, that data is also queried into a table to create what they call the *opt-out wall*. Then they try to better understand why people may be opting out of the process.

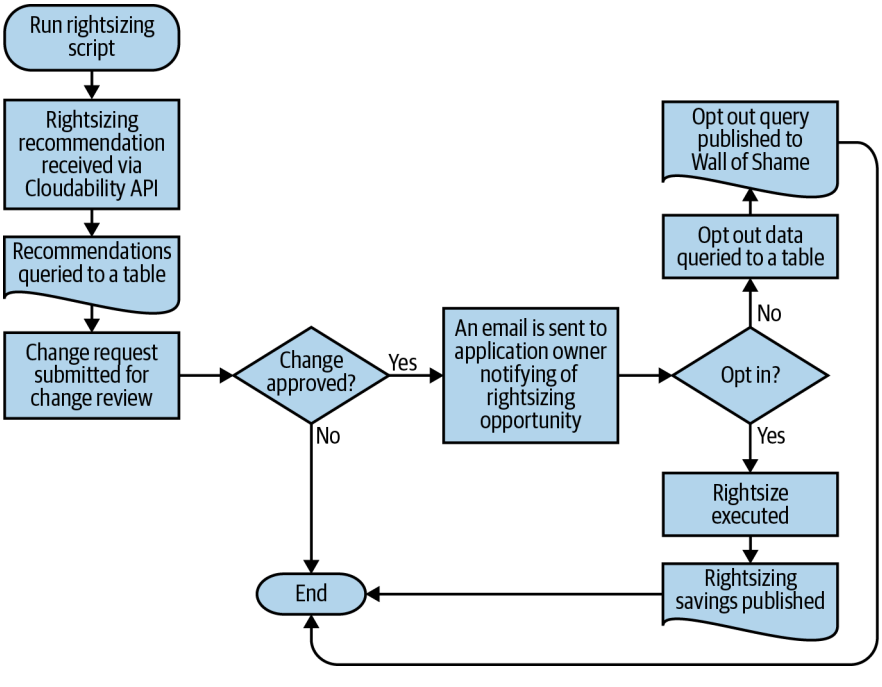


Figure 15-4. Automated rightsizing workflow

Figure 15-5 shows an example of an email an application owner might receive when they have a candidate for rightsizing, with information on the instance and the recommended change and savings.

Hello,
Your AWS Instance has been identified as a possible candidate for rightsizing.
Instance ID: i-09876543as
Name: dev-test-01
Account: 1234-5678-9901
Current Instance Type: c5.16xlarge
Recommended Instance Type: m5.xlarge
If you do not opt out of this change the FinOps Team will apply this rightsizing recommendation
`on the 23-Sept-2009.
The rightsizing process is as follows:
- Stop instance
- Change the instance type
- Start instance
To opt out please visit the following link: *https://finops.internal.example.com/rightsizing/opt-out/?InstanceID=i-09876543as*

Thanks

Figure 15-5. Sample email notification for automated rightsizing

Automating changes to an environment is an advanced FinOps process and would be recommended for mature practices. At a minimum, you should consider tracking your recommendations and having your teams manually implement the needed changes.

We talked previously about using tickets to track rightsizing recommendations, but it goes deeper than just monitoring feedback. Using a ticketing system should enable you to identify:

- The number of investigations into your recommendations
- How many of the investigations end up with savings versus no action being taken
- How long, on average, your recommendations are sitting without movement
- How many of your teams are actively performing the changes you recommend
- The type of recommendation feedback, including business justification or technical justification for not rightsizing

> *Drive tickets to teams and use JIRA to your advantage because that's how engineers do their planning. So if your optimization is in a waste ticket type X assigned to Team Y they have to plan for it or close it with punishment.*
>
> —Jason Fuller, HERE Technologies

Tickets assigned to a responsible owner always work better. Assigning tickets to team members generally makes them feel more accountable for cost wastage.

One-off automated changes or resizing of resources that are running is typically not a successful pattern. In organizations using infrastructure as code, the right way to fix it is to actually change the code and restack the environment, instead of playing a whack-a-mole game against runtime, which will just revert back to the inefficient configuration upon restacking, in the absence of code updates.

—Jason Rhoades, Intuit

Tracking Savings

Unlike rate optimization, there are no details in your cloud bill to show the discount or savings you get doing usage optimizations. Unless you have a process that enables you to directly attribute the change in instance sizes to your usage reduction efforts, the benefits of all that hard work will be difficult to track.

Usage optimization is also known as *cost avoidance*, because the only sign of savings in your bill is the lack of charges. Looking at the bill, you might notice a reduction in usage and be tempted to sum this up as savings made by your usage optimization efforts. However, just because you recommended that a team look into resizing a resource doesn't mean that they made the changes based on that recommendation. Alternatively, you might see resource usage increasing even though teams may be implementing your recommended rightsizing changes. And as if that weren't enough, any reduction in usage could be hidden by new resource usage for other projects. So trying to show the individual savings made across your cloud resources becomes very difficult.

My FinOps teams used "mini business cases" to document all the opportunities they identified for savings. These could be some sort of ticket ideally, or small document, or even a row on a shared spreadsheet. We document each opportunity, whether that's a rightsizing option, a stranded resource, a VM modernization, a license conversion, etc. We document them and track them over time to create a ledger of what opportunities we've recommended, and then ideally track which were actioned. It's a key way for a FinOps team to track and quantify their activities and justify cost avoidance claims throughout the year.

—Rob Martin, Director of Learning, FinOps Foundation

Some recommendations lend themselves easily to tracking their cost impact. For example, AWS r5 instances cost 45% less than r3 instances. Every day that you run an r3 and don't pull this trigger, you're wasting money. However, there are also important technical reasons that might preclude moving to r5 instances (the compatibility of Amazon EMR [Elastic MapReduce] versions we're running, for example), so in addition to the opportunity value, you need to know the cost of implementing it and the context of technical barriers to implementation.

Recommended changes might also have opportunity costs—requiring important team resources who are working on other projects, for example, or delaying the migration of other systems to cloud. The results of these could make the change fiscally inadvisable.

Teams can be defensive when given optimization recommendations from outside the team. We encourage FinOps teams to use these as opportunities to start conversations. You should ask questions about justifications and service use rather than presenting them as business cases. A formal business case seems like it would provide clear and constructive direction, but it can put a lot of pressure on a team to justify their past actions and selections. It might feel like you're attacking the premise or content of the business case rather than the opportunity. These are the kinds of experiences that cause teams to shy away from working with the FinOps team in the future.

For FinOps teams who meet with their teams (or top spenders) regularly, it's effective to discuss recommended optimizations in each meeting. This way, they can be raised as potential opportunities in one meeting, teams can investigate which are possible/advisable, and then they can work with the FinOps team during the month to build the mini–business case and schedule activity. In subsequent meetings, they can report on progress and can report cost avoidance for their team. This puts the savings (or the reason it's not being done) on the record in the FinOps ledger and meeting notes. It also provides information for the central FinOps team to use in evaluating commitments and discounting activities.

An alternative way to track usage optimization efforts is to look at the recommendations being made. If you sum up the total potential savings you could make by implementing all your recommendations today and then compare that figure to the amount you could have saved at some point in the past, you can determine whether your organization is getting more optimized or less so. Taking this total potential savings figure, and working out how much of a percentage compared to the total spend you have on the same resource type, allows you to determine a potential savings percentage.

For example, say you have $10,000 of total potential savings recommendations for server instances. If you currently spend $100,000 on server instances, you're operating at a 10% potential savings. If you divide the savings recommendations and total spend (using the tags and account allocation strategies discussed in Chapter 12), you can compare one team to another.

To calculate potential optimization savings, consider what the life span of the positive effect is. For instance, if you terminate an orphaned object now that saves $1,000/month, how long do you keep counting that $1,000? For the rest of that month, quarter, or year? One answer might be to consider "how long would this thing have kept going were it not for the FinOps waste program" and to consider the delta in time between when it got fixed and this length of time. These timeframes can be different across different types of waste and across teams, so there is no hard and fast rule to apply.

However you decide to calculate savings and track them over time, do it consistently, document your assumptions, and have it ready. Because one thing most FinOps practitioners report is that sooner or later the leadership in charge of the FinOps team will stop by to ask how much the FinOps team has saved by optimizing, and they usually want an answer right away.

Effective FinOps practitioners focus on the teams with the highest potential savings percentages. They assist those teams in understanding the recommendations, explain how to opt out of the recommendations when doing so makes sense, and provide expertise on the impacts of resizing resources.

Gamifying the work of optimizing, particularly for teams that are more in steady-state or maintenance modes, can be a good driver of behavior. Depending on the company's culture, it might be possible to use a worst offenders list, which uses the idea of being crowned the most wasteful to pressure teams to look more seriously into the recommendations. But negative metrics of wastefulness, or ones that call out offenders, might not be taken well. The (*Total Cost* – *Savings Opportunities*) / *Total Cost* formula is a percent-optimized metric that can be stated positively. If you build 100% optimized, your score is 100. If you don't, it's lower. You can track cumulative unoptimized cost as a tracker over time, but you want to encourage both optimization work and optimized architecture.

One of the best ways to influence positive behavior is to give each organization a savings accumulation graph that totals up the savings achieved from taking recommendations over time. Because savings can compound, teams will quickly see that the more they take rightsizing seriously, the more their savings graph will compound, giving them an impressive hockey-stick-like growth trajectory month over month. By taking 10 recommendations per month, by the end of six months, they will have the accrued benefit of 60 rightsizing recommendations. While the first few months might look like a linear growth curve, the savings numbers begin to look impressively large as they multiply over time.

Conclusion

Usage optimization is often more difficult than rate optimization, and it usually involves multiple teams working together to have the right recommendations, investigations, and changes implemented by the correct teams. The savings that can be realized by usage optimization can be significant.

To summarize:

- Usage optimization is about using only the resources needed for a workload and only when that workload needs to be running.

- Visibility into waste can lead to a leaner culture inside an organization, as teams become more aware of the impacts of waste.
- Use high-quality rightsizing recommendations to avoid stalling out any rightsizing initiatives or causing unnecessary engineering pushback.
- Bring optimization options to engineering teams with discussion and questions rather than mandates, to allow correct decisions to be made with all the facts.
- Formal workflows around usage optimization, especially when combined with automation, can lead to the best outcomes.
- Usage optimization is the hardest process to reduce cloud costs, and it should be implemented carefully using an incrementally improving approach.
- Track optimization savings closely to show the impact of the FinOps work, and work collaboratively with teams on a regular cadence to investigate opportunities, remembering that not all of them can be acted on.

Usage optimization can cause rate optimization issues due to the usage you've committed to being (re)moved based on the recommendations. It's crucial to take usage optimizations into account when performing rate optimizations to avoid making commitments on usage that will change.

Now that we have covered optimizing usage, we'll proceed to rate optimization, where you'll save further by reducing the rate you pay for cloud resources.

CHAPTER 16

Paying Less: Rate Optimization

In Chapter 5, you learned that your *Cloud Cost* = *Rates* × *Usage*. The previous chapter looked at how to manage the *usage* part of the equation—using resources in the size you need, and only when you need them. This chapter and the two that follow focus on the other half of that equation, and cover how to optimize rates to pay less for the resources you continue to use.

This work will primarily be managed by the centralized FinOps function for your organization, which has the highest-level view of your organizational cloud usage, the specialized skills and knowledge to purchase these specialized discounts, and the responsibility to manage your cloud rates across the business.

By now, you know that a dollar is not a dollar when paying for cloud. With multiple purchasing options, billing in different time and volume increments, a slew of payment structures and commitment options, and unique differences from each cloud provider, customers have a panoply of purchasing options, each of which has its own unique financial ramifications. This chapter covers the basics of the pricing options you can use to manage your rates. Reservations, Savings Plans (SPs), Reserved Instances (RIs), Committed Use Discounts (CUDs), and Flexible CUDs are the primary levers for adjusting rates for many services, but they can be quite complex, so we talk about those specifically in Chapter 17. Then, in Chapter 18, we talk about strategies you can use to bring all of your rate optimizations tools together in a holistic way.

Compute Pricing

Rate optimization and usage optimization both typically begin by looking at compute usage—what we're paying for the AWS EC2 instances, Google Compute Engine, Azure virtual machines, managed containers, or serverless usage that we're running in cloud. There are two key reasons for this. First, for most customers, compute costs

are the largest single cost category, often by far. And second, compute pricing is one of the oldest and most mature offerings from cloud providers, and it has the most options for obtaining rate discounts based on commitments.

Depending on the cloud service provider and the resources you're using, cloud resources can be priced many ways. Different cloud service providers also use different terms for similar pricing offerings. First, let's take a quick look at the big three providers in Table 16-1.

Table 16-1. Comparison of reservation options across the big three cloud service providers

	AWS	Google	Azure
Public price	On-demand	On-demand	Pay-as-you-go
Spot	Spot	Spot/preemptible	Spot VM
Sustained use discount	N/A	Sustained Use Discounts (SUDs)[a]	N/A
Reserved	RIs/SPs	CUDs/Flexible CUDs	Reserved VM instances/SPs
Volume discounts	Volume discounts	Volume discounts	Volume discounts

[a] SUDs are being phased out for new instance families, because customers favor the predictability of CUDs.

On-Demand/Pay-As-You-Go

When you request a normal resource and run it without making any precommitments (like reservations), you're charged the list price, or the on-demand/pay-as-you-go rate. This is typically the highest price you can pay for a resource. It's also the most flexible, as you can stop using it at any time without a penalty.

Spot Resource Usage

Most cloud service providers offer a way to run a resource using the spare capacity of their cloud platforms. With spot pricing, you typically set a maximum price you're willing to pay for the resource, which can sometimes be more than the on-demand rate but usually is set much lower. You will pay that spot rate for the spot resource until your price threshold is exceeded or there is no longer available spot capacity to fulfill your request, at which point you will have a short period in which to stop using the resource before it's recalled by the cloud service provider. These low rates are possible as opportunistic utilization of what would otherwise be idle cloud capacity.

This pricing offers significant savings (up to 91%!) over on-demand rates and normally can be the cheapest method of running resources. However, the risk of losing the resource at any minute means the services running on spot resources need to be able to handle the reduction of resource availability. Common uses of spot resources are for batch processing that can be resumed or restarted if you lose the resource.

Commitment-Based Discounts

Commitment-based discounts is a good general term for rate discounting options like Reservations, Reserved Instances, Savings Plans, Flexible CUDs, or Committed Use Discounts. Generally speaking, these types of discounts allow you to make a long-term commitment to the cloud service provider to run a set amount of usage of (or spend a certain amount on) a particular type of cloud resources—such as compute, databases, AI/ML (artificial intelligence/machine learning)—for a period of time (sometimes in a particular place or with particular parameters). In return, the cloud service provider will discount the normal on-demand rates for that set amount of usage or cost. The more specific your commitment or the longer you commit or the more you pay up front, the more your discount will be in most cases. As stated earlier, Chapter 17 covers details about commitment-based discounts in great depth.

Storage Pricing

In addition to Azure Storage reserved capacity to reduce the rate you pay on storage, all the major cloud service providers have a range of storage services that provide different levels of availability, durability, performance, and capacity. Generally speaking, the more performant, durable, and available your data is, the more you pay. The key is to find a balance between the price, availability, and quality of the data storage. Generally we would consider that a usage optimization, but in this case we are referring to the differences between the rates you pay for different tiers of storage rather than reducing the amount you store in order to cut costs.

You should optimize your storage tier based on the access patterns of your stored data. This can save significant amounts without the need to reduce storage volume. Each of the major cloud providers offers various, and frequently evolving, storage tier options to choose from, depending on whether you frequently or infrequently access the data.

Consider production data that's being served to customers. This will most likely need to be stored on the most performant, highly available storage offerings. But a backup of this data is less likely to need storage on the same higher-cost offering. As your backups age, they're not likely to be needed in a pinch, so the data could then be stored in the lowest performance level with low availability, but still durable, storage layers. When you move data like this, it's called *data lifecycle management*.

Some cloud service provider storage offerings can manage the lifecycle of your data for you. With S3 in AWS, data can be relocated to lower-availability storage tiers based on rules. For example, after 1 month, S3 can move the data to lower available

storage, and after 12 months, it can be moved to a very cheap "cold" storage offering called Glacier, which can take hours to retrieve your data.

The more effort you put into classifying and locating your data in the different service offerings, the more you'll save on data storage costs. Using the built-in features of the cloud services to automate the data lifecycle can reduce management overheads. It is, of course, very important that your data be stored on the correct service offering with the amount of availability and durability you require.

Volume/Tiered Discounts

Most cloud service providers' built-in pricing plans for some of their services will reduce cost as you use more. When you have large amounts of usage, these automatic price reductions can add up to substantial savings. Generally, these volume discounts are either usage-based or time-based.

Volume discounts may apply at a regional level, meaning usage across regions won't be combined for discounting. Consider possibilities to consolidate cloud resources into a smaller number of regions to take full advantage of volume discounts.

Volume discounts reward large cloud users, ensuring that growing on a cloud service provider's platform continues to make sense for large enterprises. To calculate projected costs correctly on the cloud platform, you need to identify the type of volume discount you're getting. When calculating savings in reducing your usage, you must understand which pricing tier applies to the usage being reduced. Using the rates from the correct pricing tier enables you to calculate costs (and possible savings) correctly.

Keep in mind also that volume discounts usually apply to usage over the specified volume; they typically do not alter the rate paid for usage below the discount tier.

Usage-Based

When a discount is applied based on the total amount of usage rather than the length of time a resource is running, we call it a *usage-based volume discount*. The amount of usage in a single hour is combined, and then a rate discount is applied to the amount of usage over the tier thresholds.

When you look at Figure 16-1, which shows your usage-based volume discount in action, the pricing tiers run on the y-axis (usage). When usage exceeds the Tier 2 threshold (within any set hour), all usage above that threshold will be charged at the Tier 2 pricing. Usage discounts are applied each hour using this same calculation.

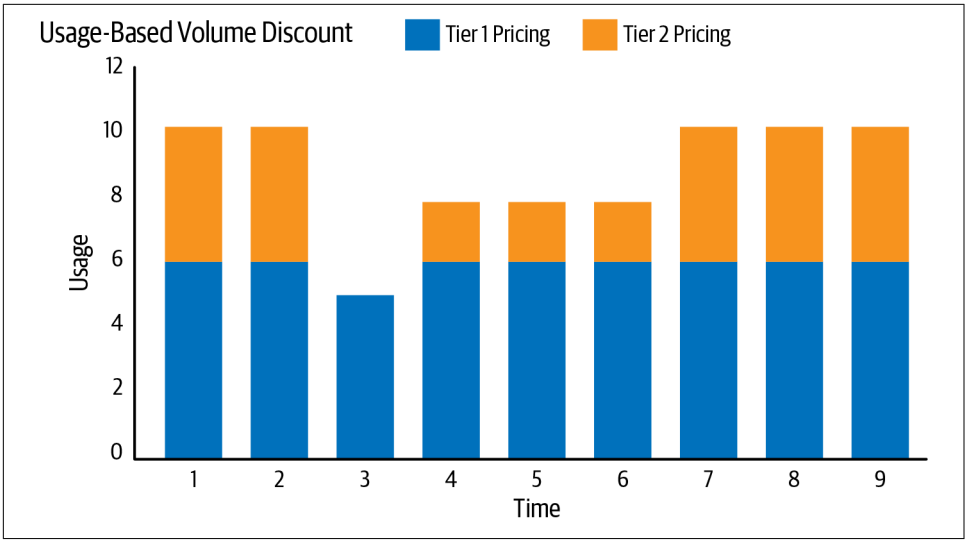


Figure 16-1. Usage-based volume discount graph

A *free tier* is a special type of volume discount that applies to a small amount of cloud usage. Cloud service providers offer these free tiers on many of their services to enable customers to try the service with lower costs. Some free tier offerings are available only for short periods of time, while others are available on an ongoing basis. For organizations with large cloud environments, the savings generated by the free tiers are often immaterial.

Time-Based

When a resource is charged at a lower rate after it's been running for a particular amount of time, it's called a *time-based volume discount*. A good example of this type of discount is the Google Cloud Sustained Use Discount (SUD), where compute usage is charged at lower rates the longer it is running. However, it's worth noting that Google Cloud SUDs are being phased out. Some time-based discounts apply to a specific resource and stop applying as soon as the resource is removed, while others match a resource usage type and can be applied to multiple resources over time. Note that time-based volume discounts aren't combined with other resources running at the same time. For example, 100 resources running in the same hour are not equivalent to one resource running for 100 hours.

In Figure 16-2, you can visualize how time-based volume discounts work. In the first hour, 10 resources are being used, and in turn 10 individual volume discount counters are started. Let's assume the cost is charged at Tier 1 pricing for the first five hours of usage, and then at Tier 2 pricing after that. The important change from the usage-based scenario is that the volume discount applies on the x-axis (time), so even

though you have 10 resources in the first hour, they all are charged at the Tier 1 price. As each hour progresses, the 10 timers are started and stopped based on the amount of resource usage.

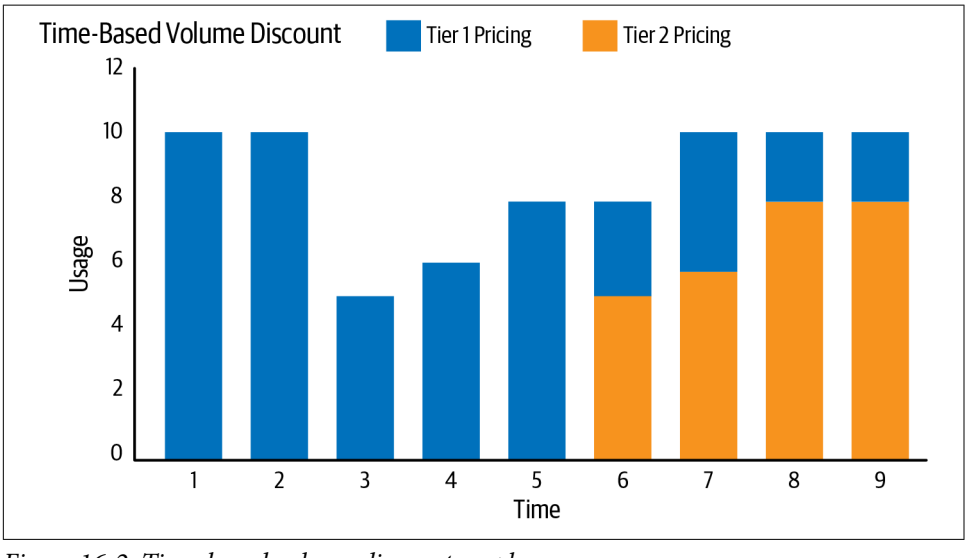


Figure 16-2. Time-based volume discount graph

For Resource 1—which runs all the time—you receive the Tier 2 price after five hours in the graph. For Resource 6, however—as it stops and starts over time—it takes eight hours on the graph for it to reach Tier 2 pricing.

Negotiated Rates

In some situations you may be able to negotiate with the cloud service providers, cloud resellers, or marketplace vendors for better pricing.

Custom Pricing

For large usage levels of some services, the pricing page published by cloud vendors advises you to call them. At these top usage tiers, the cloud service providers might be willing to give you better pricing than they offer publicly. We're unable to go into any details on what these deals are, as they often come under nondisclosure agreements.

When you have outgrown the published volume discount tiers, be sure to start conversations with your cloud providers early to determine if there are any opportunities for deeper discounts, either across the cloud bill in total, in a region, or at a service level.

Seller Private Offers

When you're using third-party software in the cloud, license agreements must be organized, and licensing payments are usually made directly to that third party. The license is then applied to cloud servers to run the software. This leads to having one bill from your cloud service provider for the resources to run your software and a separate bill from your software vendor for the licenses. With the creation of services like AWS, Google Cloud, and Azure marketplaces, third-party vendors can now offer their software through these channels. The license and resource fees are paid directly on the customer's cloud provider bill, and the cloud provider, in turn, pays the license fees back to the third-party vendor.

While the marketplace reduces the effort to calculate the total cost of running services, you now pay a fixed public rate for the license costs. When you organized licensing directly with your third-party vendor, you were able to negotiate better rates for your organization. Fortunately, some services, such as AWS Marketplace, now allow vendors to have Seller Private Offers. Customers negotiate the license rates with the third party, and a custom marketplace listing is created specifically for that customer via a private link. When an organization uses these services direct from a marketplace, they pay custom rates for the license and maintain the single billing source convenience direct from the cloud provider.

BYOL Considerations

Some cloud service providers allow you to use existing license agreements that you might already have within your organization to reduce the costs of cloud, which is known as the BYOL ("bring your own license") model. One example of this is the Azure Hybrid Benefit for Windows Server. Organizations with an existing Software Assurance or Subscription License (such as the Enterprise Agreement Subscription [EAS]), Server Cloud Enrollment (SCE) subscription, or Open Value Subscription on their Windows Server licenses are able to reuse the license(s) when they deploy servers into the Azure platform.

Identifying opportunities to reuse existing licenses—especially while migrating from a data center into the cloud—avoids unnecessary licensing costs, which can save organizations large amounts of money.

Use cases for owned/perpetual licenses or license rights purchased for noncloud to be used in the cloud are extremely complex and are often subject to changing rules and requirements. If your organization uses a significant amount of licensed software or maintains a large number of agreements with other software vendors that you intend to use in the cloud, we strongly recommend considering the implications of BYOL and involving professionals with procurement or software asset management

perspectives to advise on how to use these rights most effectively and within your rights.

Conclusion

By varying service performance, availability, and durability, cloud service providers are able to offer similar services at different rates. Effective FinOps teams centralize this function and keep engineers focused on using the service offering that best fits the profile of their workloads. Using usage optimization and rate optimization together can produce significant savings.

To summarize:

- Rate optimization is about getting charged lower rates for the same resources.
- Compute pricing can be reduced by using spot, committing to usage over time, achieving volume discount levels, negotiating discounts, or applying licenses you already own.
- Storage pricing can be reduced by selecting the appropriate tier of service to meet your needs.
- For large usage amounts you are able to negotiate better rates, either directly with your cloud service provider or with third-party software license vendors.

It's important to realize that there are multiple ways to reduce the rate you're charged for the same cloud resources. Next up, we'll cover the most popular and arguably the most complex rate reduction mechanisms: using Reserved Instances, Savings Plans, Committed Use Discounts, and Flexible Committed Use Discounts.

CHAPTER 17

Understanding Commitment-Based Discounts

Commitment-based discounts are a perfect example of the collaborative, data-driven decision making of FinOps. They bridge the gap between technology teams and finance teams, requiring alignment that can be challenging to achieve. However, when alignment is reached, commitment-based discounts can bring a significant cost efficiency advantage in the form of improved cloud unit economics, giving you the same compute power for a big discount.

This chapter is one of the most likely to become out of date—probably before this book even makes it to market—as there are always new updates to cloud provider offerings. We suggest checking the details in this chapter against what your cloud service provider is currently offering. Fear not, though: while there have been many changes to offerings over the years, the core best practices around reservations remain consistent.

Introduction to Commitment-Based Discounts

Reserved Instances (RIs), Savings Plans (SPs), Committed Use Discounts (CUDs), and Flexible Committed Use Discounts (Flexible CUDs), collectively known as commitment-based discounts, are the most popular and important cost optimizations that cloud service providers offer. This is because commitment-based discounts represent the largest percentage discount you can achieve in cloud and often apply to the largest areas of cloud spend in your bill.

This chapter focuses primarily on commitment-based discounts for compute services. There are discount offerings on other cloud services, including databases and AI/ML. Compute costs are often the largest area of spend for most organizations. For these other discount offerings, some of the concepts here can be applied, but you should check with the cloud service provider's documentation.

Each cloud service provider has different offerings with its own specific rules on the discounts it provides. And each organization can take advantage of the various commitment-based discounts in different ways based on the implementation models and business processes of the unique organization.

Commitment-based discounts are awesome. You commit to using a particular type of resource for a certain amount of time, and you get a considerable discount. The idea seems simple. But as you'll see, there are many nuances to learn.

Some years ago, during a webinar with AWS, Cloudability, and Adobe on the power of RIs, Adobe showed Figure 17-1, indicating that the company had cut its EC2 spending by 60% simply by purchasing RIs.

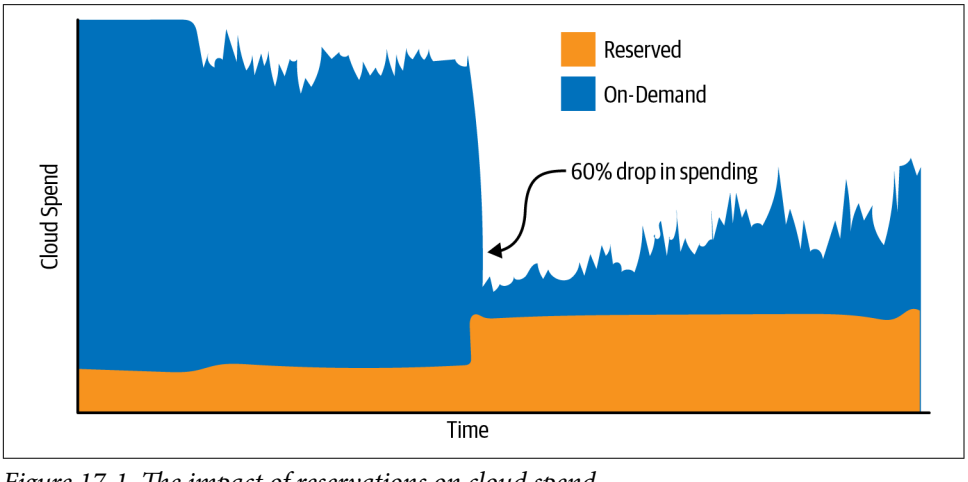


Figure 17-1. The impact of reservations on cloud spend

By moving about 10% of their money from paying on-demand rates to paying the discounted reserved rates, Adobe was able to increase its overall *reservation coverage rate* and achieve this dramatic 60% reduction in the overall cost of their compute resources. Success stories like Adobe's make the decision to utilize reservations/commitments an obvious one. However, for many organizations, getting from no RI coverage to good coverage is a process that takes a lot of organizational education and alignment.

Stories from the Cloud—J.R.

I worked with a FTSE 100 UK retailer that had millions of pounds a month of cloud spending with AWS, but zero RIs. They knew that buying even a modest amount of RIs would save a lot of money. In fact, working closely with their team to analyze the company's infrastructure, I found they could save millions over the coming three years. Because they're a retailer, margins were tight, and the RI purchase could help them achieve critical cost reduction goals mandated by leadership.

However, despite all of this, it still took them a full nine months to buy their first RIs. What happened? Why was the organization so slow to adopt such an obvious cost-savings measure despite aggressive top-down cost-cutting goals? The answer is because they had to educate—and align—many stakeholders in the organization around what buying RIs meant and, in the process, dispel many misconceptions cemented during years of data center hardware purchasing.

First, they had to get their finance team on board with how to financially account for RIs. Up-front payments for RIs can appear to be capital expenditure (CapEx) spending that finance should depreciate as a physical asset due to their up-front payments. However, RIs are intangible prepayments of operational expenditures (OpEx) that need to be amortized over the period they're used.

Next, they had to get their technical teams comfortable with committing to current infrastructure. The strength of cloud lies in your ability to use only what you need and to adapt infrastructure to fit the shape of your workload. Then you introduce new approaches like serverless or containerization and the promise (or threat) of completely refactoring infrastructure on a new set of cloud services. The risk of changing the infrastructure makes teams hesitant to commit to their current stack for the one year or three years of a reservation. Foot-dragging by these teams, who were grounded in the admirable plan to improve their infrastructure, caused multiple delays in purchasing reservations.

Stories like this are sadly common. As companies implement their first reservations, getting all teams on board can take a long time, and the lost opportunity of savings can add up to millions of pounds. Had this retailer purchased some RIs as they educated the business during that nine-month period, they would have realized significant savings—and avoided millions in unnecessary on-demand spending.

Commitment-Based Discount Basics

While the specifics of each cloud service provider's offerings are different, all of the big three provide you with the same simple ability to commit to a particular resource usage over a set period. Your commitment ensures continued business for the cloud service provider, and in return they reward you with a lower rate for the usage (rate optimization). The discounts can be quite impressive—in some cases up to 80%.

In all three primary cloud providers, you don't actually reserve a specific server instance or commit to a specific resource. Remember, you've left behind the data center concepts of naming servers. Discounts are applied at billing time by the cloud service providers. In fact, reservations are essentially a billing construct. Some AWS RIs can offer some capacity reservation functionality outside of the billing discounts—in fact AWS Reserved Instances originally *only* provided capacity reservations, not discounts.

Commitment-based discounts are most often applied to individual resources in a nondeterministic way. In the case of AWS SPs and Azure SPs, they are applied where they will have the biggest savings, but you can't pick which specific resources they will be applied against. Google Cloud lets you choose how to attribute the discount credits and fees. They give you three options: *Unattributed*, *Proportional Attribution*, or *Prioritized Attributions*. The right attribution model to choose depends whether your organization purchases and manages CUDs in a centralized or decentralized way.

Cloud service providers apply these discounts based on a loose set of rules, and the details of why and where they go are fairly opaque. Beyond prioritizing based on the highest savings potential in the case of SPs, the providers will allocate the reservation randomly to resources in the account, subscription, or project where they were purchased, and then any leftover coverage will flow to eligible resources in other accounts, subscriptions, or projects (unless you tell them not to). Randomly applying the discounts introduces many strategy and chargeback considerations that we'll discuss in Chapter 18. With cloud service providers that supply resource-level billing data, you might not see the discount being applied directly to your resources, or at least not to the resources you were expecting.

An analogy might be useful here to help you better understand reservations/commitments. Say a specific restaurant is running a deal where you buy a book of coupons. Each coupon gives you a meal at that specific restaurant. The book contains one coupon for every day of the month. When you eat at the specified restaurant, you pay for the meal with the coupon. Deciding to eat somewhere else means that you forfeit that day's coupon and pay full price on the meal at another establishment.

Let's say the book costs $750 and contains 30 meal coupons, where each coupon gets you a meal that, if bought without a coupon, would cost $50. Divided out, that's $25 per coupon for a $50 meal, saving you $25 a day. If you eat at this restaurant every day, you save 50%, and if you eat there only half of the days, you've saved nothing. If you use more than half of the coupons, you're better off buying the book.

If you apply this idea to RIs, once you've decided on the length of time you want to reserve, you purchase a reservation (book of coupons) from the cloud service provider, matching a particular resource type and region (specific restaurant meal at a certain location). This reservation will allow you to run the matching resource

every hour (or second or millisecond). If you don't run any resources matching the reservation, you forfeit the savings. As long as you have enough resource usage during the reservation term, you benefit from the discounts—and you save money.

The key takeaways here are:

- You pay for the commitment-based discount, whether it's applied to a resource or not.
- Reservations cost money, but they offset the cost of resources in your account.
- You do not need to utilize a commitment-based discount fully to save money.

The specific purchase options of a commitment-based discount and what resources they can apply to are different for each cloud service provider. Some offerings apply to very specific usage; others apply widely.

Across the whole range of commitment-based discounts, you can choose many options:

- An amount of *usage hours* (Reservations, CUDs) or *hourly spend* (SPs/Flexible CUDs)
- Any of a variety of *compute* resources running anywhere
- A specific type of *compute* or one of a list of other resources
- With the right to convert to a different type of reservation or not
- Running in a particular region or location
- In a specific operating model (dedicated or shared)
- Of a specific size (or of a flexible size—instance size flexibility)
- Running a specific operating system or database type
- In a very specific availability zone
- For one or three years (or even five years, for some services)

Generally speaking, the more specific your commitment, the higher your discount will be. In other words, you can achieve higher savings if you're willing to give up on flexibility. Over time, the cloud service providers have introduced a number of levels of flexibility, which we cover generally here and in more detail in the sections on each provider.

Compute Instance Size Flexibility

Cloud service providers offer hundreds of different server instance types or sizes, typically grouped into families, which you can sort out from the name of the virtual machine (VM) or instance.

Examples of instance types from various providers:

- *AWS instance m5.xlarge:* An "m" family instance of size "xlarge" in the "5th" generation
- *Azure instance standard_D2_v5:* A "D" series VM of size "2" in the "5th" generation
- *Google Cloud instance n2-standard-128:* An "N2" family instance of size "standard" with "128 CPUs"

Many commitment-based discounts are purchased to match a particular resource type's usage parameters. This means you would need to make potentially hundreds of specific commitments to cover all the various sizes of instances you use, even if you use the same family of instances widely. To make this process easier, cloud service providers offer *instance size flexibility* (ISF) on some of their reservations.

Within a specific family, you can choose from a range of server instance sizes (different amounts of vCPU and memory), with each increase in size being proportionally more expensive. For example, on AWS, two matching large instances are the same price as one extra-large instance of the same type (e.g., two m5.large instances cost the same as one m5.xlarge instance). So it makes sense that two purchased large RIs can apply their discount to one extra-large server instance, and vice versa.

The one caveat to this is in the area of software license fees: two medium instances are not the same as one large instance if there are software license fees involved. With two mediums, you need twice the licenses than you'd need for one large. So AWS excludes ISF from servers with licensed software, while Azure charges you separately for the software license costs with reservations (applying a discount to the compute hours only). Note that when you pair reservations with Azure Hybrid Benefit, the software license costs are also zeroed out.

Due to the way Google Cloud charges, they naturally do ISF by combining all instances' vCPU and memory counts within the same machine family. CUDs apply to the combined total instead of against the individual discrete instances.

ISF is very helpful for engineering teams who wish to use a certain family of compute or database services but want the flexibility of changing the size of the instances they are launching as the needs of their workloads change. ISF gives these teams much more flexibility and dramatically reduces the workload on the FinOps team because no action is required to move the coverage to different instance sizes—it happens automatically.

Conversions and Cancellations

Some commitment-based discount programs offer the ability to convert commitments or exchange them for another type. AWS Convertible reservations offer lower discount rates than reservations that cannot be exchanged, but allow users the option of exchanging the reservation for a different family or a different size (if ISF does not apply). The ability to convert reservations lowers the risk of ending up with reservations that don't match your resource usage. If resources get updated so they no longer match, you can exchange the reservation for one that does. The ability to exchange or convert reservations is typically restricted to the region in which they were originally purchased.

Keep in mind also that converting RIs from one type to another requires analysis and work on the part of the FinOps team, so consider this cost in the overall equation when considering your options.

It should be noted here that AWS SPs, as of this writing, offer the same discounts that AWS RIs offer, but with convertibility built in. A compute SP, for example, covers compute instances in multiple regions and in any family without requiring any analysis or conversion to modify them, at the same discount as a convertible RI in one region for one type of family. SPs are discussed further in the section on AWS.

Azure does not offer convertible RIs, but it does allow users to cancel and exchange a certain dollar value of reservations each year. Sometimes this entails a fee, and usually it also requires the new reservation to start the clock over again from the time of the exchange. Azure SPs are now available as an alternative to reservations and function similarly to AWS SPs, offering much more flexibility than Azure RIs.

Google Cloud offers the ability to split or merge commitments. Splitting allows you to redistribute resources across your projects or organization. Merging commitments makes it easier to manage the discounts as a single entity with one expiration date. Google Cloud also now offers Flexible CUDs, bringing similar flexibility as SPs at AWS or Azure.

In general, the conversions you make must be equal or greater in value and duration. So if you lower your usage of the cloud service provider, you can't exchange reservations if doing so results in a lower overall commitment.

One additional option AWS users have is the ability to sell certain types of reservations with a significant amount of time remaining on a secondary marketplace. Typically this can only be done with Standard, or nonconvertible, RIs with more than half their life remaining.

Overview of Usage Commitments Offered by the Big Three

Table 17-1 shows the big three cloud providers and compares their offerings.

Table 17-1. Comparison of reservation offerings across the top three cloud service providers

	AWS	Google	Azure
Program name	Reserved Instances/Savings Plans	Committed Use Discounts/ Flexible CUDs	Reservations/Savings Plans
Payment model	All up front, partial, no up front (different discount rates)	No up front	All up front, monthly (same price)
Term	One or three years	One or three years	One, three, or five years
Instance size flexibility	Yes	N/A	Yes
Convert or exchange	Standard, No Convertible, Yes	Yes, splitting and merging	Yes
Cancelable	No; however, some RIs can be sold via a marketplace	No	Yes, up to a yearly value limit
Sustained use discounts	No	Yes[a]	No

[a] SUDs are being phased out for new instance families, because customers favor the predictability of CUDs.

You can see that the big three cloud service providers' offerings are very similar, with similar term length and size flexibility. Where the discount offerings differ is largely driven by their specific methods of applying the reservation and method of billing for the discounted cloud resources.

Amazon Web Services

AWS was the first to offer RIs back in 2009. Thus, they have by far the most complex reservation programs. However, this also means that AWS is the most flexible and configurable. Reservations are available for EC2 instances, RDS databases, Redshift nodes, ElastiCache nodes, OpenSearch nodes, and DynamoDB reserved capacity.

DynamoDB reserved capacity is different from the other reservations. With DynamoDB, you reserve a set amount of reads and writes to this service. The remaining reservation types each have slightly different features, but what's common to all of them is that you purchase them in one- or three-year terms. You can pay either nothing up front, make a partial payment up front, or pay all up front for the reservation, with each increase in up-front payment giving you a few more discount percentage points. When purchasing RIs, you buy one or more with the same parameters in what AWS calls a *lease*.

You can think of the difference between the total amount you pay for a resource covered by an RI using partial or no up-front payment models and the all up-front payment model as the interest AWS charges you to lend you the up-front payment. By figuring this out (it's different for each resource's RI) you can compare this imputed interest rate to your internal cost of capital and see whether it's better to invest the cash in buying up front or taking the lower discount and paying less (or nothing) up front.

RIs are confusingly named, since you aren't actually reserving a specific instance. Nor can a reservation be guaranteed to apply to a specific resource. Remember, RIs are more like coupons you purchase and apply against relevant resources nondeterministically.

RIs are often thought of as a percentage discount applied to your resource's cost. However, it is also correct to say the RI has a fixed rate that's a certain amount cheaper than the on-demand rate at purchase. While this might sound like the same thing, it's important to understand that if AWS reduces the on-demand rate, your existing RI rates remain the same during the one- or three-year term.

Price drops may affect your decision about which RI configuration to purchase—we'll get into that later in the chapter. It's worth noting that an analysis of historic AWS price drops showed that AWS doesn't typically reduce the rate of on-demand enough to undercut the three-year RI price.

What Does an RI Provide?

AWS reservations have two parts: a rate reduction benefit and in one case, a capacity reservation for the type of resource purchased in a specific availability zone. The majority of RI purchases are made to get a rate reduction. But some companies also want a capacity reservation as well, ensuring that the number of resources they reserve of a particular type in a very particular place are available when they want to run them.

Over the years, we've seen the capacity reservation become a less important aspect of RI purchasing as AWS has scaled their infrastructure. However, some customers running massive, elastic infrastructures still optimize for capacity as well. More recently, AWS and Azure have offered On-Demand Capacity Reservations (ODCRs), which allow you to perform capacity reservations separately from RIs. Google Cloud offers a similar concept called Compute Engine zonal resources, which provide a very high level of assurance in obtaining capacity for a specific VM type. Note that when you're using on-demand capacity reservations in combination with AWS RIs, it's essential to set your RIs as regional so the RIs can discount the capacity reservation.

AWS Commitment Models

Let's look more closely at the various forms of commitment-based discounts that AWS offers.

We'll go into more detail on how AWS commitments work, as many of the same concepts and models carry over to the other cloud providers, and Amazon has the most maturity and complexity in this area currently.

AWS Reserved Instance

You buy AWS RIs for a specific resource type in a specific region. AWS will match the RI *coupon* you purchase to relevant resources based on these criteria when your bill is generated.

The primary parameters are:

Region
: Where is your resource located (e.g., US-West-2 or EU-West-1)?

Scope
: Is your reservation zonal or regional? If zonal, it will match usage in a particular availability zone (e.g., US-West-2A). If regional, the RI will apply to any availability zone in a region (e.g., US-West-2A or US-West-2B). Note that only zonal RIs receive a capacity reservation, but one can be purchased for regional RIs.

Availability zone
: If a zonal RI, to which specific should the RI apply (e.g., US-West-2A)?

Instance characteristics

  Type
  : Consists of the family of the instance (e.g., m5, c4, t3) and the size. Instance sizes vary per family, but they generally have values like large, xlarge, 2xlarge, and so on.

  Tenancy
  : Determines whether the resource is shared or dedicated.

  Operating system
  : Specifies which operating system and software licenses are included with the instance (Red Hat Enterprise Linux, Microsoft Windows, etc.).

If your RI parameters don't precisely match the usage you're running, then the RI won't apply a discount. So it's vital to ensure you're buying the reservations with the correct parameters.

AWS will nondeterministically apply the discount to a randomly selected resource, like coupons applied against relevant resources. While ISF will apply from the smallest to the largest instances, there's no current way to ensure that the discount is applied to a specific resource instead of any other matching resource.

Member Account Affinity

One way customers can influence how RIs get applied is to use member account affinity. AWS Organizations provides a feature called *consolidated billing* that allows for accounts to be a member and costs to be consolidated into a single management account (previously called the payer account). There can be only one management account within an AWS Organization, and accounts can be members of only one AWS Organization. But of course you might have multiple management accounts in your business for various reasons.

When you purchase RIs, you can perform this purchase in the management account or in any member account. RIs may share the discount they offer across accounts within the same AWS Organization. However, they have an affinity to the local account they belong within, which means the RIs purchased in a specific account will first be made available to the resources in that account. This works the same way for SPs, which we discuss shortly, but we'll just use RIs in this section to describe how account affinity works.

If the local account doesn't have any matching usage for an RI, then the reservation will apply a discount to matching usage in any other AWS account that is part of the same AWS Organization.

RI sharing can be disabled on a per-AWS-account basis, but it's on by default. If RI sharing is disabled, any RIs within that account will not apply a discount outside of the account, and when the RI doesn't match any usage within the account, it won't apply any discounts, something in FinOps we term *vacancy* or reservation wastage. Disabled sharing effectively pins an RI to a member account if it's critical for budget purposes that it not be shared. Consider when you disable sharing it could result in waste, as discounts won't be shared to others when not used. This doesn't seem to make any sense, and it usually doesn't, but it might be required in cases where money is authorized only for a specific purpose, such as in a governmental agency where funding is very restricted in its use. Also, RIs in other accounts in the same AWS Organization will not apply discounts to any accounts that have RI sharing disabled.

People commonly misunderstand how RI sharing works. To ensure you get it right, take a look at the example in Figure 17-2.

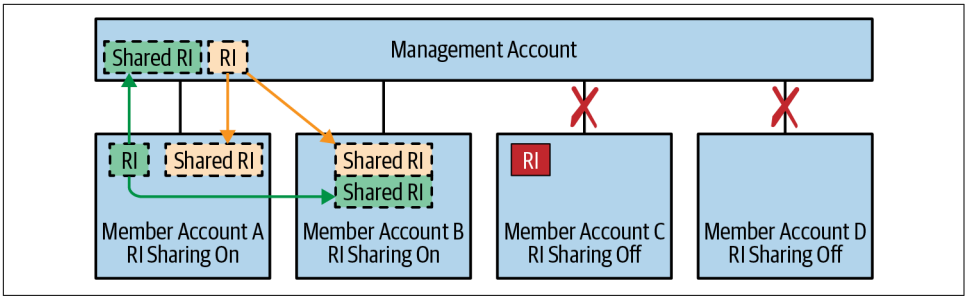


Figure 17-2. Discount interactions with RI sharing

Figure 17-2 shows a typical account structure: a management account with a handful of linked member accounts. You can see in the diagram there are three RI purchases: in the management account, in member account A, and in member account C. Account affinity will cause the RIs within the account to apply a discount to the local account first. If no usage matches the RI within the same account, then RIs may apply a discount to usage in another member account via RI sharing.

In the example layout in Figure 17-2, the following may occur:

- RIs in the management account can apply to account A or B because they both have RI sharing enabled.
- Account A can benefit both from the RIs purchased within the account and from RIs in the management account but will have preference for the RIs within the account.
- Account B can benefit from RIs from both account A and the management account.
- Because account C has RI sharing disabled, the RIs within that account will not apply a discount outside of account C and will go unused when no usage matches.
- Account D will not receive any RI discounts because it has RI sharing disabled and does not have any RIs purchased within the account.

It's worth taking a moment to reread this list. The majority of people we've met over the years commonly misunderstand some part of RIs, and this can lead to RIs not applying discounts where you expect them to. Knowing how to control where RIs apply and where they don't allows you to choose the best strategy for where to buy reservations.

Standard Versus Convertible RIs

When you buy an RI from AWS, you can purchase multiple RIs at once as part of a single lease. A *Standard Reserved Instance* (SRI) allows you to set the parameters as you initially buy them.

With an SRI, it's possible to modify some of these parameters, such as availability zone and scope (regional versus zonal). For Linux/UNIX RIs, part of the same RI lease can be split and joined (e.g., two mediums become a single large). However, you can't modify many other parameters, such as term length, operating system, instance family (m5, c5), and tenancy (shared or dedicated). Once a standard RI is purchased, you are committed to running resources that match these parameters for the RI term (one or three years).

For EC2 instances, a more flexible option, the *Convertible Reserved Instance* (CRI), is available. In return for a lower discount percentage, you can exchange RI reservations for different ones during the term. Some limitations still apply to exchanges.

No matter what, you can't reduce your overall commitment to AWS. For example, you can't swap an expensive RI reservation for a cheaper one if the exchange results in a lower commitment. In that case, a true-up charge is required that makes the modification include more RI or more expensive parameters. When an RI exchange results in more expense, AWS will charge you the net difference in any up-front payment due.

There's a process that allows you to split CRI reservations so that you can convert only some portion of the RIs instead of all the RIs within a single RI lease. Similarly, there is a process to join multiple CRI reservations together, allowing you to modify them all into one new CRI reservation. However, modification of some parameters of the CRI isn't possible during the term:

- CRI leases are purchased for a particular region and cannot be exchanged for another region.
- CRI leases are for EC2 instances only and cannot be converted to other services; for example, if you move your database from an EC2 instance into RDS, you cannot exchange your CRI for an RDS RI.
- The term for CRIs cannot change, except that when joining CRIs you can sometimes extend the term from one to three years.

Besides these limitations, overall CRIs have a much lower risk for an organization. Unless you're likely to reduce your total usage of EC2 during the term, you can change your EC2 usage parameters during the term and exchange your CRIs for reservations that match your parameters. Keep in mind, however, that the analysis

to figure out which conversions to make and the effort to manually perform the conversions still require time and effort by the FinOps team.

EC2 RIs also contain a feature called *capacity reservation*. RIs configured as zonal (i.e., they apply to resources in a particular availability zone) will contractually guarantee server capacity. This capacity is available only to the same AWS account as the RI and applies only to the specific resource configuration in the availability zone selected. If you have unutilized RIs in the account, and a request to AWS for an EC2 instance matching that reservation is received, you can guarantee the availability of capacity. Just like the RI discount, you cannot allocate the capacity reservation to any particular instance, so the first matching instance to run within that account will consume this capacity reservation.

EC2 RIs set as regional will apply a discount to any resources in any availability zone in that region, but it will not provide any capacity reservation. Regional RIs can, however, be combined with a related AWS service, On-Demand Capacity Reservations (ODCR). ODCR instance reservations are charged at on-demand rates, like a running EC2 instance, but can also be covered by the RIs you own to achieve the discounts of RIs across a region with capacity reservations.

Instance Size Flexibility

EC2 RIs are purchased to match a particular instance size (small, medium, large, xlarge, etc.). The RI will apply discounts to resources matching its size. However, for regional Linux/UNIX RIs, you benefit from the feature mentioned earlier, called *instance size flexibility* (ISF). Reservations you currently own or are planning to buy with attributes of Linux, regional, and shared tenancy will automatically be an ISF RI.

ISF allows the RI to apply discounts to different size instances in the same family (m5, c5, r4, etc.). A single large RI can apply discounts to multiple smaller instances, and a single small RI can apply a partial discount to a large instance. ISF gives you the flexibility to change the size of your instances without losing the discount applied by an RI. Because you don't need to be specific about the exact size of the RI for it to cover all your different size instances, you can bundle up your RI purchases into small variations of parameters.

Figure 17-3 illustrates a way of thinking about how ISF can be applied. Each column represents the instances that were run in a given hour. For r5 and c5 instances, large (L) is the smallest instance size. If you aim for 100% RI utilization in this example, you'd purchase 29 large RIs. Think of it as purchasing LEGO blocks at the smallest size within a family and combining them to cover larger instance sizes you are actually running. This would mean you'd have only one size of RI you purchase that's well matched to your overall usage. If instance sizes fluctuate but normalized usage stays the same or increases overall, your RIs will remain perfectly utilized.

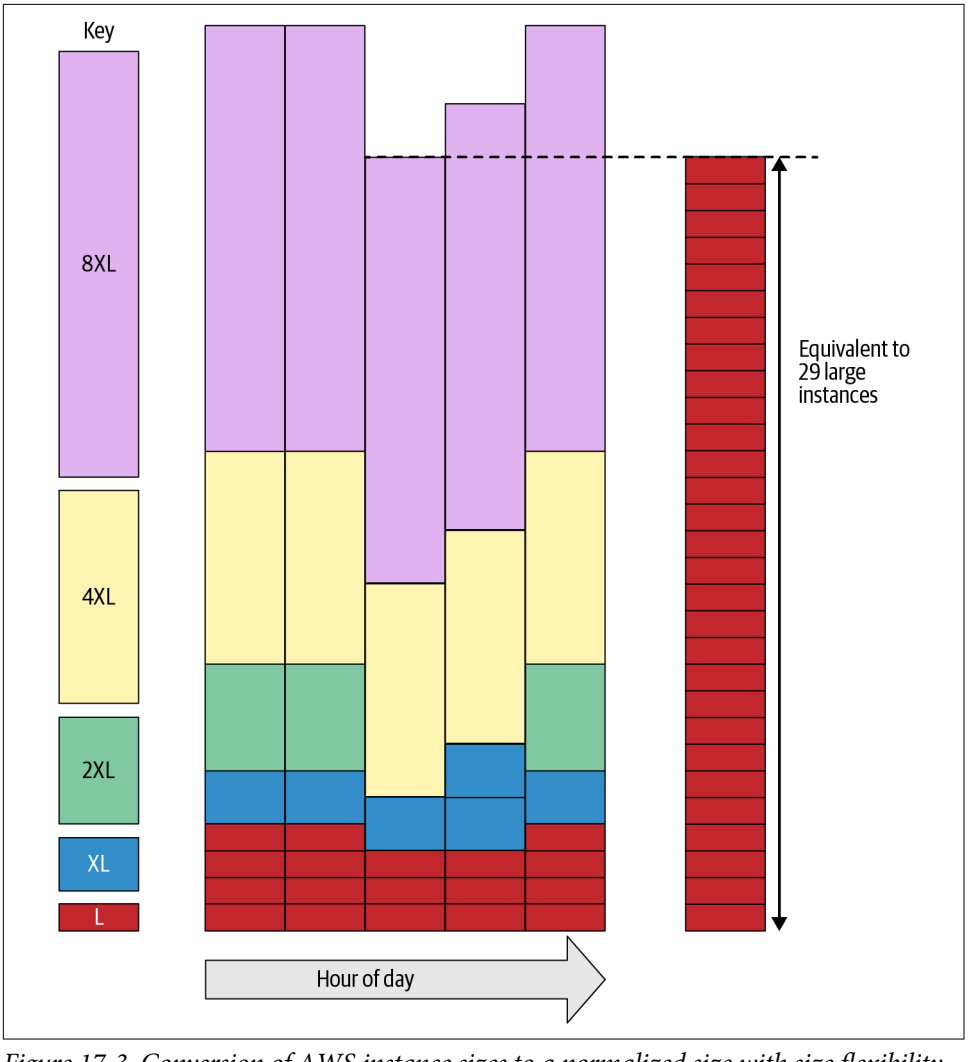


Figure 17-3. Conversion of AWS instance sizes to a normalized size with size flexibility

AWS uses a normalization factor to determine how a particular RI is applied with ISF. Using it makes it possible to work out how many normalized units a particular size instance has. An RI uses this same normalization factor to determine how many normalized units of discount it can apply per hour.

The normalization factor in Table 17-2 shows how to convert between instance sizes within a family. For example, if you own an RI that's 2xlarge (16 units), that RI could apply instead to two xlarge (8 units each) or four large (4 units each) instances. You could also use an RI that is 2xlarge (16 units) to apply to one xlarge (8 units) and two large (4 units each) instances.

Table 17-2. Normalization factors for AWS reservations

Instance size	Normalization factor	Instance size	Normalization factor
nano	0.25	6xlarge	48
micro	0.5	8xlarge	64
small	1	9xlarge	72
medium	2	10xlarge	80
large	4	12xlarge	96
xlarge	8	16xlarge	128
2xlarge	16	18xlarge	144
3xlarge	24	24xlarge	192
4xlarge	32	32xlarge	256

The billing for ISF RIs depends on what size instance you run relative to the RI you own. In Figure 17-4, you have four small reservations, which will cover four small instances. If you don't have any small instances with ISF, the reservations would apply to two medium instances. Without any medium instances, you can cover a single large, half of an xlarge, or a quarter of a 2xlarge instance. The larger instances can be covered by a smaller size RI, and you'd get a prorated on-demand charge for the remainder of the units uncovered for that instance size.

Some instance families do not have all the sizes of instances, so, if purchasing ISF RIs, just look for the smallest instance size in the family.

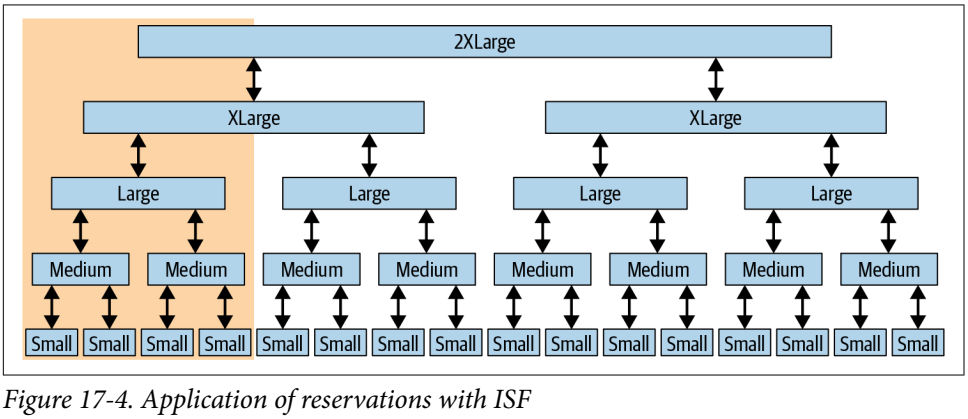


Figure 17-4. Application of reservations with ISF

AWS Savings Plans

In late 2019, Amazon Web Services announced Savings Plans, initially offering discounts on EC2 instances, Lambda, and Fargate, the managed service of Amazon's proprietary container service. Since the initial release, AWS has added an additional SP offering that applies to the AWS machine learning service SageMaker. The addition of the machine learning SP has driven up demand from AWS customers for AWS to release further SPs to cover other services, such as database and storage. Based on this customer demand, we expect additional offerings to be announced after this edition of the book is released. A large amount of what you learned here about RIs applies to SPs as well. AWS has continued the purchasing options of *All upfront*, *Partial upfront*, and *No upfront* payment and one- and three-year durations.

However, the biggest difference between SPs and RIs is that while RI commitments are for resource units (numbers of EC2 instances of certain sizes), SP commitments are monetary (the amount a customer is committing to spend on discounted compute or other covered services).

SPs are offered in three plan types:

Compute SPs
: Apply broadly across EC2, Lambda, and Fargate compute, offering savings comparable to CRIs. This plan type applies more widely than a CRI—to include compute resources in any region—which will lower the amount of effort in maintaining high plan utilization. These generally return the same savings rate as CRIs.

EC2 Instance SPs
: Apply to EC2 usage of a single family in a single region in any size or other configuration. While this plan type is more restrictive than a Compute SP, it offers higher discounts and is less restrictive than an SRI, while providing the same discount as SRIs.

Machine Learning SPs
: Sticking with the cost/hour model of SPs, the machine learning SP for SageMaker enables AWS customers to commit to spend on SageMaker in return for reduced rates on component costs of SageMaker, including those on:

  - Studio and On-Demand Notebooks
  - Processing
  - Data Wrangler
  - Training
  - Real-Time Inference
  - Batch Transform

The SP will automatically apply to SageMaker usage across any AWS region and instance compute family.

There are, however, a few major differences between RIs and SPs:

- Both SP types that discount EC2 instances offer more flexibility in the compute they apply discounts to compared to RIs, removing the need to consider tenancy, operating system, instance family, or instance size. Most importantly, Compute SPs apply across all regions, which will significantly improve how much of your usage is coverable.
- With RIs, you purchase a number of reservations that match instances in your accounts. SPs are purchased as a committed hourly spend. The spend amount that you commit to is post-discount. For example, if you're running $100/hour of on-demand EC2 and an SP offers a 50% discount, you need to purchase a $50/hour SP.
- Unlike RIs, SPs apply discounts to AWS Fargate and Lambda usage. This is the first AWS offering to discount usage across multiple service offerings. Because these other services are discounted at very different discount rates, this also can affect the overall benefit you receive from an SP, discussed more in the next chapter.
- There is no equivalent RI offering for AWS SageMaker.

AWS will apply SP coverage to the resources that give you the greatest discount, which is nice because not all types of compute or instances are discounted by the same amount. This also means that, as you get more and more coverage, the increase in discount percentage you are receiving will get smaller.

Savings Bundles

In early 2021, AWS introduced a new concept they've called a CloudFront Security Savings Bundle. This is a new type of commitment-based discount that covers usage of CloudFront, AWS's content delivery service, and allows customers to commit to a level of hourly spending on the service in exchange for a 30% discount on the CloudFront service and credits for use of WAF, AWS's web application firewall service. Customers commit to discounted spending levels, so this is very similar to an SP, but the bundling of free WAF usage makes it a different type of benefit leading to a new marketing name for the discount.

There may be additional services that are often used together, which may get similar Savings Bundles in the future. Again, all of these commitment-based discounts from AWS are billing constructs only, so they give AWS the ability to reward regular committed use with discounts in a way that recognizes these usage patterns and rewards those customers who are able to forecast effectively and commit to longer-term use.

This is a great reason to get good at forecasting and to get your engineering teams involved in providing their future plans so that the FinOps team can get the best rate optimization coverage possible without overpurchasing.

Microsoft Azure

Azure offers both reservation and SP vehicles.

Azure Reservations

Azure's reservation options largely followed the model of commitment-based discounts that AWS offered, but with some additional flexibility and convertibility or cancellation options.

As of late 2022, Azure reservations are available for 27 different services and resource types, which is a great amount of coverage. Services include VMs; blob, disk, and file storage; databases, like SQL and Cosmos DB; analytics services, like Azure Synapse Data Explorer; Azure App Service; Azure VMWare; Red Hat and SUSE Linux; and many more. The list is constantly growing, so please refer to the Azure reservations documentation for an up-to-date list.

Some services, like Azure Databricks, use reserved capacity slightly differently than other reservations. With Azure Databricks, you reserve a set amount of Databricks Units (DBUs) that are then consumed throughout the entire term. You aren't limited to a strict hourly commitment.

Azure reservations are purchased to cover usage in a specific Azure region (when applicable), SKU or size, and term (one, three, or five years). Operating system and availability zone are not as specific for Azure VM reservations as they are in some AWS offerings.

Azure RIs can be exchanged across any region and any series as a workload or application needs to change. Note that compute reservations cannot be exchanged after January 2024. If you need an option to exchange compute reservations, start with a savings plan. Noncompute reservations will continue to support exchanges. A prorated refund is given, based on remaining hours, to be applied to the new RIs that are part of the exchange.

Azure RI exchange targets must be equal in value to, or of greater value than, the source RI. There's no concept of a "top-up," but if the new reservation is of greater value, some additional investment over the credited amount will be required. In addition, an "exchange" can simply be a cancellation of the entire RI purchase with a fee.

It's currently possible to cancel Azure reservations for an adjusted refund if the capacity purchased is no longer needed. The fine print does state that cancellations have a 12% early termination fee, and cancellations max out at $50,000 per year.

It has been announced that exchanges and refunds on Azure reservations for compute services will no longer be offered after January 2024; the introduction of the more flexible Azure SPs helps offset this change.

There is a capacity prioritization, but no guarantee. This means when there is insufficient capacity to satisfy a request for resources, those with a reservation will be provided capacity before others who don't have one. Without the guarantee, however, it's possible that a request could be denied if Azure is out of capacity of that particular resource type.

Azure RIs can be assigned at an Enterprise Agreement billing account (enrollment), Microsoft Customer Agreement billing profile, management group, subscription, or resource group level, so you can manage RI usage at an organizational, departmental, or application level. This gives you more control—compared to other cloud provider offerings—over how to apply RIs and who should benefit from them. If the goal is to save money organization-wide, then RIs can simply be assigned at the billing account or billing profile level.

If a particular business unit wants to buy the reservation for its use only, the RI can be assigned to a management group, subscription, or resource group where only that group can take advantage of the savings. The downside to this feature is that unused RIs don't become available to be used by machines not in a subscription, which could result in more waste than with AWS, where RIs automatically cascade to other member accounts if the purchasing account doesn't use them. The designation of RIs at the enrollment or subscription level can be changed as necessary after purchase.

Azure VM reservations cover compute costs only and do not cover software license fees. Since on-demand (nonreserved) usage covers both compute hours and software license fees, reserved usage will generate an extra record for the separate software license fees when applicable. Note that Windows Server, Red Hat Linux, SUSE Linux, and SQL Server license fees can also be covered for additional savings by using Azure Hybrid Benefit to assign on-premises licenses to the VM or SQL instance based on the number of cores running.

Azure RIs are available for all VM families other than A-series, A_v2 series, or G-series but are not offered in all regions for the VM families. Check Microsoft's documentation to verify that the RIs you wish to purchase are available for your VMs.

Azure reservations can be purchased with either up-front or monthly payments. The up-front payment option works like AWS's *All upfront* option, with a single payment for the entire term at the beginning. The monthly payments are processed on the same day each month and are similar to AWS's *No upfront* option; however, unlike AWS, there is no difference in the discount percentage between paying up front or not.

Instance Size Flexibility

Azure ISF works almost exactly the same as AWS: just substitute the Azure VM "size groups" for "instance families," and "ratio unit" for "normalized unit."

Instances within the same size series group can have a reservation applied, no matter the size of the reservation. A smaller reservation size can apply a partial discount to a larger instance, and a larger reservation can discount multiple smaller instances. Each size series group has a table of ratios for each specific size instance. Again, it works exactly the same as AWS's normalized units. Using these ratio units, you can determine how many units you need to cover your VM instances.

Let's look at an example to help clarify how this works (see Table 17-3).

Table 17-3. The ratio table for the DSv3-series instances

Size	Ratio units	Size	Ratio units
Standard_D2s_v3	1	Standard_D16s_v3	8
Standard_D4s_v3	2	Standard_D32s_v3	16
Standard_D8s_v3	4	Standard_D64s_v3	32

Now, if you look at VM instance usage over a period of time for this specific size series, you can determine how many ratio units are being used during this period (see Figure 17-5).

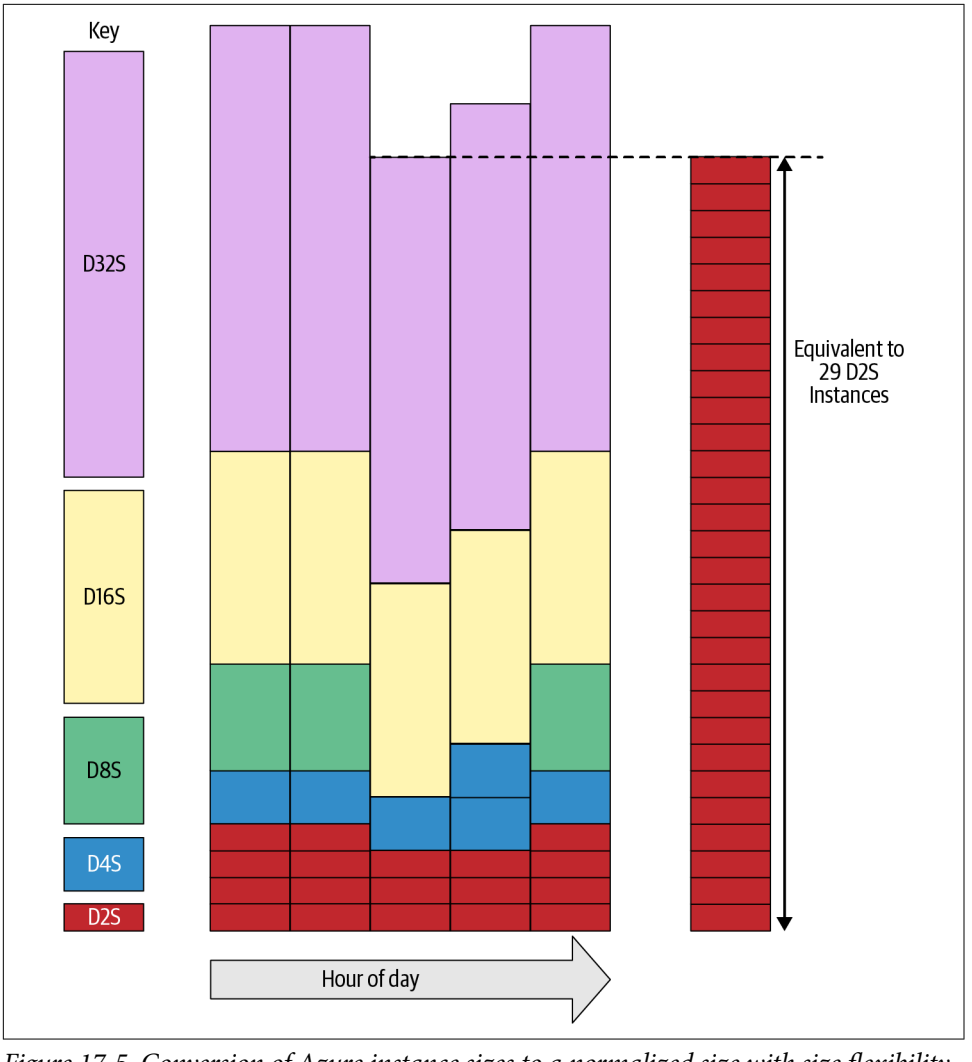


Figure 17-5. Conversion of Azure instance sizes to a normalized size with size flexibility

You can see from Figure 17-5 that you're using 29 ratio units of VM instances 100% of the time, and based on this unit count, you can determine how many reservations you need to make.

Azure Savings Plans

Azure SPs operate as discounts to usage hours over a period of time. You save money by committing to a fixed hourly cost on compute services over a one- or three-year term. Azure advertises savings of up to 65% from pay-as-you-go prices. Discounts

vary based on the commitment term (one or three years), not amount of hourly spend commitment.

The impact an Azure SP has on your hourly spend depends on the amount of spend within the hour and commitment you have made. Figure 17-6, found on the Azure documentation page for SPs, demonstrates how a plan will apply in particular situations:

Hour 1
: The usage exactly matches that of the savings plan amount, which results in all usage getting the discounted rate.

Hour 2
: The usage is under the commitment amount and the *unused savings paln* will be charged on top of the discounted usage. Meaning you pay the committed hourly rate.

Hour 3
: The usage is more than the commitment amount. Usage up to the committed amount is discounted and usage over the committed amount will be charged at *pay-as-you-go* prices.

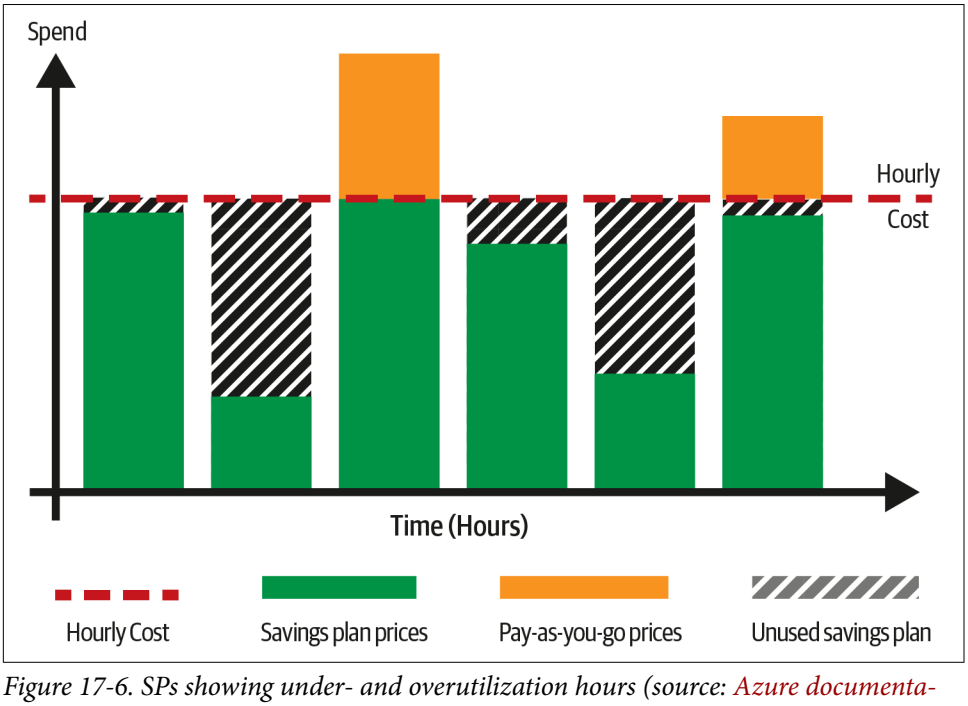


Figure 17-6. SPs showing under- and overutilization hours (source: Azure documentation)

Each hour, your compute usage is discounted until you reach your commitment amount; all subsequent usage is priced at pay-as-you-go rates.

SP commitments are priced in USD for Microsoft Customer Agreement and Microsoft Partner Agreement customers, and in local currency for Enterprise Agreement customers.

It is possible to trade in one or more of your active reservations for a new SP. When you perform a reservation trade for an SP, the reservation is canceled and the prorated residual value of the unused reservation benefit is converted to the equivalent hourly commitment for the SP. It is not possible to reduce the value of the SP below that of the residual value; however, you can increase it to cover your needs.

After you purchase an SP, the discount automatically applies to matching resources. Like reservations, SPs provide a billing discount and don't affect the runtime state of your resources.

You can pay for an SP up front or monthly. No matter if you pay up front or monthly, the SP has the same cost; you don't pay any extra fees when you choose to pay monthly.

Azure SPs cover usage from compute services such as VMs, dedicated hosts, container instances, Azure premium functions, and Azure App Services.

Google Cloud

Google Cloud offers Committed Use Discounts (CUDs), a Google Cloud billing construct that offers deep discounts in exchange for committed usage, and Flexible CUDs, which we'll discuss later in the chapter.

Google Committed Use Discounts

According to Google Cloud, CUDs are available in one- or three-year terms, with discounts of up to 57% for most (including custom) machine types, and up to 70% for memory-optimized machines.

CUDs offer several degrees of flexibility in how they are purchased and consumed. First, CUDs are not limited to any one particular compute instance or shape in your account but rather applied toward the overall eligible aggregate usage of a particular resource type in a region throughout the month. CUDs apply to aggregate resources across different VM shapes, operating systems, and tenancy. In addition, a new offering called Flexible CUDs even allows the commitment to extend across multiple VM families and regions. CUDs are also applicable across resources across all zones in a

region, which allows flexible cross-zone deployment models while taking advantage of the savings that CUDs offer. Commitments are for a specific region and a specific project. They can also be shared across projects in a billing account, if you enable CUD sharing in your account. Recall from Chapter 12 that projects organize all Google Cloud resources into logical groupings. Zone assignment within a region isn't a factor like it is with other providers.

Paying for Cores, Not Hours, in Google

This might seem obvious, but the fact that Google Cloud doesn't charge for VM instance hours is a significant difference from other cloud vendors, and it has a number of implications. The VM product in Google is called Google Compute Engine (GCE), and its SKUs are individualized to CPU cores, RAM, and localized disk. This undoubtedly is rooted in the fact that you can create a completely custom machine with any ratio of CPU cores and memory to your heart's desire. Here are some of the implications:

- When Google exposes resource-level info in the billing data, you should expect the instances to have multiple rows for each hour corresponding to the different parts. This will provide the ability to roll up to instance-level costs.
- When purchasing CUDs, you're making a commitment specifically to CPU cores or memory (but not local storage). Because purchases are made at a ratio of your choosing (within a range), there is a very good chance that an even number of underlying instances will not be covered. For example, it's possible to cover five instances' worth of CPU cores but only four instances' worth of memory. As a result, CUD planning should be done independently for cores versus memory.

Having this clear divide makes container resource and cost allocation simpler, since the underlying components are already split out. By contrast, when the underlying cost is a VM instance (as in the case of AWS), then special math is needed to split the cost out.

Google Billing and Sharing CUDs

Let's look at how CUD billing and application works within the Google concept of organizations, folders, and projects. Figure 17-7 shows the structure of the hierarchy.

There are similarities to AWS's concept of a management account structure. A billing account can share CUDs across all of its connected projects, once you've enabled CUD sharing on the account. This means CUDs offer both the flexibility of applying to a variety of machine types and the flexibility of applying to multiple projects.

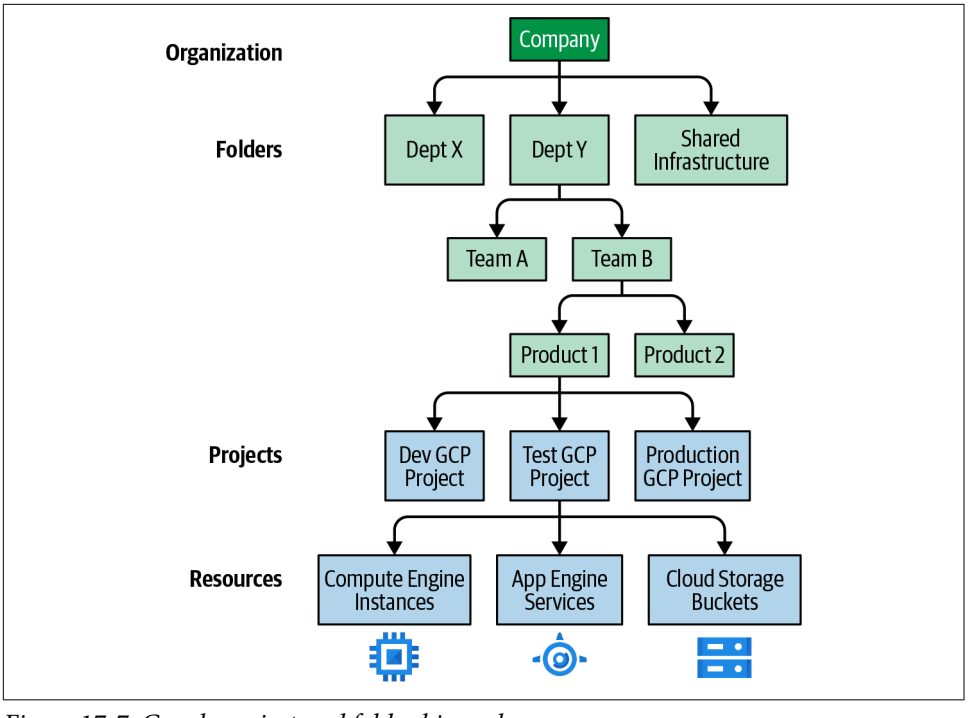


Figure 17-7. Google project and folder hierarchy

Unlike AWS, where RIs are automatically shared across multiple linked accounts, CUDs are shared only after you've enabled the feature for your billing account. This gives you the flexibility to choose what's most important for your account: cleaner chargeback or maximum waste reduction. If you want to allocate funds from a particular project to make the commitment, turning CUD sharing off will offer cleaner chargeback options than the way AWS handles RIs. However, turning CUD sharing on could result in less waste because unused commitments will be shared across multiple projects.

CUDs give you the option to choose the sharing model that works best for your organization. For an organization with a lot of projects, turning sharing on allows you to take advantage of the economies of scale of the entire organization.

Google Billing Account and Ownership

Two types of relationships govern the interactions between billing accounts, organizations, and projects: ownership and payment linkage.

Figure 17-8 shows the ownership and payment linkage relationship of projects. *Ownership* refers to identity and access management (IAM) permission inheritance from the main organization. *Payment linkage* defines which billing account pays for a given

project. In the diagram, the organization has ownership over projects A, B, and C, and the billing account is responsible for paying the expenses incurred by the three projects.

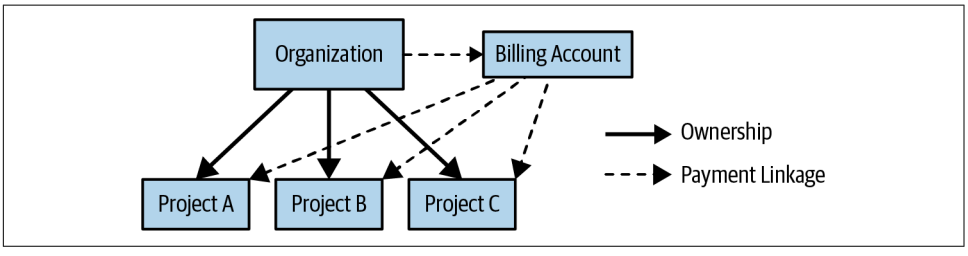


Figure 17-8. Google billing account and ownership

Applying Google CUDs in a Project

CUDs can't be used to cover short-term bursts in usage. If you purchase CUDs for 50 vCPUs, you can't run 100 vCPUs for half of the month and receive the full discount. Revisiting the coupon analogy, a single *coupon* cannot be applied to multiple resources at the same time.

CUDs are purchased for a specific instance family, and one type of CUD can't be applied to the other resource usage type. If you buy N2 (General Purpose) CUDs, they will not apply a discount to M1 (Memory Optimized) instances, and vice versa.

You purchase CUDs in one- or three-year terms, which by default don't autorenew. If you want a commitment to automatically renew, you must manually enable the autorenew setting on each commitment. CUDs are applied to the sum total of vCPU and memory usage. A commitment for 4 vCPUs can apply a partial discount to an instance of more vCPUs. For example, an instance run with 16 vCPUs will receive 4 vCPUs at the CUD rate and the remaining 12 vCPUs at the normal rate.

Finally, it's worth noting that CUDs are allocated to the most expensive machine types in a very specific order:

1. Custom machine types
2. Sole-tenant node groups
3. Predefined machine types

This ensures the best possible discount at the time of application.

Google Flexible Committed Use Discounts

During the final stages of creating the second edition of this book in late 2022, Google Cloud released a new commitment offering called Flexible Committed Use Discounts

(Flexible CUDs). A large amount of what you learned here about CUDs applies to Flexible CUDs as well.

Similar to AWS SPs, Flexible CUDs for GCE are spend-based commitments that provide large discounts off list price in exchange for committing to a certain level of hourly spend on GCE for a one- or three-year term.

Flexible CUDs add two dimensions to the flexibility already offered by standard CUDs. As with standard CUDs, you can change the size and zones you use for your instances. First, Flexible CUDs also let you change the instance family among most general purpose and compute optimized families. Second, they let you change what region your VMs run in.

Flexible CUDs are ideal for customers with predictable cloud spend needs or who need to have flexibility to commit their cloud dollars but don't want to be restricted to one region or one machine type. Flexible CUDs allow you to decouple cloud investment financial decisions from workload-specific VM choices.

Customers commit to a consistent amount of spend, measured in dollars per hour of equivalent on-demand spend, for a one- or three-year term. In exchange they receive a discounted rate on the applicable usage covered by their commitment.

You can purchase CUDs from any billing account, and the discount can apply to any eligible usage in projects paid for by that billing account. While many of the CUD techniques in these chapters will help you evaluate your Flexible CUD decisions, it's too early for there to be specific best practices for Flexible CUDs. Stay tuned to the FinOps Foundation's Google-related forums to continue the conversation around the latest best practices from fellow practitioners.

Conclusion

RIs, SPs, CUDs, and Flexible CUDs are the most complex optimizations available to cloud users. There are serious discounts to be gained by operating a successful reservation program.

To summarize:

- All three of the major cloud service providers offer the ability to make commitments in return for large discounts. Each has differences in features, how they work, and the discounts they provide.
- To reap the most benefit from reservations, you need to understand all features, including size flexibility and what modifications can be applied after a purchase.
- Due to all this complexity, you will receive the best results if you centralize the management of RIs, SPs, and CUDs so that teams can focus on reducing their usage and not worry about rates.

- Educating various stakeholder teams such as engineering and finance is an important step in the process, as RIs are frequently misunderstood.
- Because RIs, SPs, CUDs, Flexible CUDs, and other discounts are in exchange for commitments, a clear and accurate forecast of future, consistent usage of resource types you wish to cover is essential. This requires a good working relationship between engineering and the FinOps team.

Now that you know how reservations work, we'll move on to strategies and considerations for correctly and optimally implementing reservations inside your organization.

CHAPTER 18

Building a Commitment-Based Discount Strategy

Now that you understand the fundamentals of commitment-based discounts as described in the previous two chapters, it's time to ask some tough but important questions:

- How much should you commit?
- What should you commit?
- When should you commit?
- Who should be involved in the process?
- How do you know your commitments are being fully utilized?
- How do you know when to repurchase them?
- Who should pay for them?
- How do you allocate the costs and savings over the commitment term?

Unfortunately, there are no definitive answers—only the answers that are right for a company at a particular time. Those answers may change considerably as you move through the FinOps maturity curve, as the cloud providers update their ever-changing commitment models, and as your company's cash reserves change.

Common Mistakes

We see companies make a number of errors when working with commitment-based discount programs. In no particular order, these are:

- Delaying making commitments for too long
- Making commitments that are too conservative
- Committing based on the number of unique instances instead of a waterline (covered later in this chapter)
- Not managing commitments after purchasing them
- Buying the wrong types of commitments

Nearly everyone gets some part of their early strategy wrong. That's OK. This is the early maturity stage. You can think of it like learning to ride a bike or writing your first bits of code—it's a learning process that may take a few tries to get it right.

> *Failure sucks but instructs. Discovery of the moves that work well is always accompanied by discovery of moves that don't.*
>
> —Bob Sutton, Professor of Management Science at Stanford University

Steps to Building a Commitment-Based Discount Strategy

There are six key steps in building your first commitment strategy:

1. Learn the fundamentals of each program.
2. Understand your level of commitment to your cloud service provider.
3. Build a repeatable commitment-based discount process.
4. Purchase regularly and often.
5. Measure and iterate.
6. Allocate up-front commitment costs appropriately.

This section covers each step in turn.

Step 1: Learn the Fundamentals of Each Program

Good news! Thanks to Chapter 17, you're mostly up to speed on the fundamentals of common vehicles like Reserved Instances (RIs)/Savings Plans (SPs) and Committed Use Discounts (CUDs)/Flexible CUDs used to discount compute.

However, there are a couple of concepts we didn't cover. The first is the break-even point of a commitment program. If you remember nothing else from this section, never forget this:

You don't need to use a 1-year commitment for a whole 12 months to save money.

Yes, you are committing to paying for that period of time, but you don't need to actually utilize the commitment for every hour of that commitment period for the purchase to make financial sense.

A three-year commitment can seem scary for even the most committed cloud user. In some ways it goes against everything you love about cloud: just-in-time, scalable, on-demand usage. But one- or three-year commitments to a cloud provider can save massive amounts of money and completely change the economics of cloud in your favor.

A 1-year commit may break even in as little as 4–6 months, and a 3-year commit may break even in as little as 9–12 months. Sharing data on these break-even points in your organization can help ease fears of commitments to your cloud of choice. Let's break down the various components of the commitment break-even point.

Components of the commitment break-even point

Figure 18-1 shows you the accumulated cost of a resource, for both on-demand rate and commitment rate, over a one-year term. In the example, the commitment costs over $300 up front, which is why the commitment line starts above $300. If you had used a *No upfront* commitment, then it would begin below the on-demand line. And if you used an *All upfront* commitment, it would be drawn as a flat line at the value of the up-front cost.

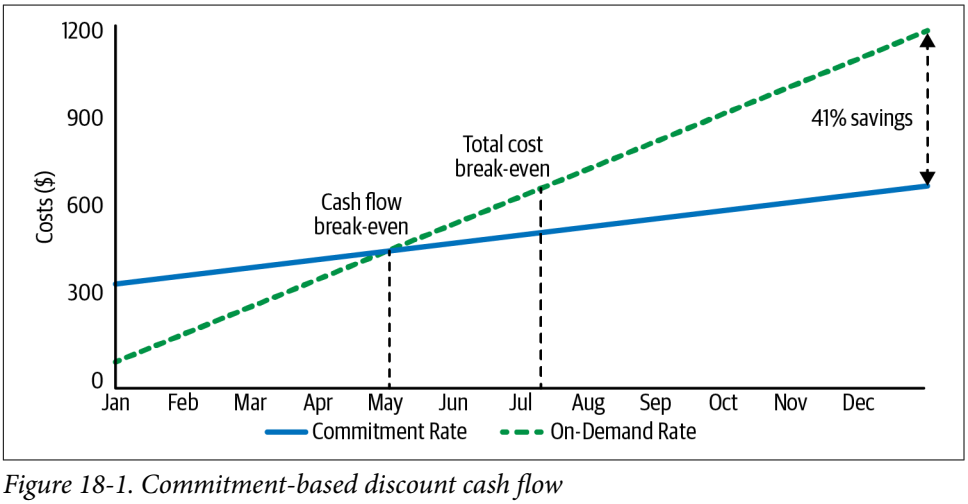


Figure 18-1. Commitment-based discount cash flow

The *cash flow break-even point* is the date on which the commitment has cost you the same amount as if you had been using on-demand rates. Some people call the cash flow break-even point the *crossover point*, for obvious reasons. You're no longer

out-of-pocket for the commitment at the cash flow break-even point. However, you will continue to pay the ongoing (hourly) costs of the commitment. If you stop your usage at the cash flow break-even point, it will result in you losing money versus on-demand rates.

The *total committed cost break-even point* is where the on-demand cost of resource usage is more than the total cost of the commitment. This is the real break-even point, since you could cease to run any usage that matches the commitment and you would be no worse off than if you had not committed to the discount program. You would have paid the same amount on-demand versus with the commitment. After the break-even point, you realize savings from the commitment.

The difference between the on-demand line and the total cost break-even point is the savings (or loss) made by the commitment. It's essential to understand that with this graph it doesn't matter whether you run one resource that matches the commitment for the whole 12 months or you run many different matching resources during that period.

If you add up all the usage to which your commit applies a discount, you can then compare the total cost of the commitment versus what you would have paid for the same amount of usage at on-demand rates. This comparison will give you an indication of when you have met your break-even point and of the amount of savings you have realized.

The majority of the time, a well-planned commitment-based discount program will break even sooner than you expect. When it comes to your strategy, the real challenge is typically less around deciding what to buy first and more around getting your organization aligned on how to buy the commitments.

The commitment waterline

Before we move on to that process, there's another fundamental concept we didn't cover in the previous chapter: the commitment *waterline*. As you know, you don't buy a commitment for a specific instance, nor can you determine where the commitment discount will be applied. The commitments are applied nondeterministically to a resource that matches the criteria of the commitment-based discount you have selected.

If you're successfully utilizing the elasticity of the cloud, you will have differing amounts of resource usage throughout the day/week/month—less during your quiet times and more during your busy periods.

Figure 18-2 shows some resource usage over time. The units in this figure are less important than the concept. For AWS RIs, the units might be individual running instance hours of instances that match the type of RI we're considering buying. For

SPs, the units might be discount-applied dollars or money per hour we're spending each hour. But they are all interchangeable units of commitment for a commitment we're considering.

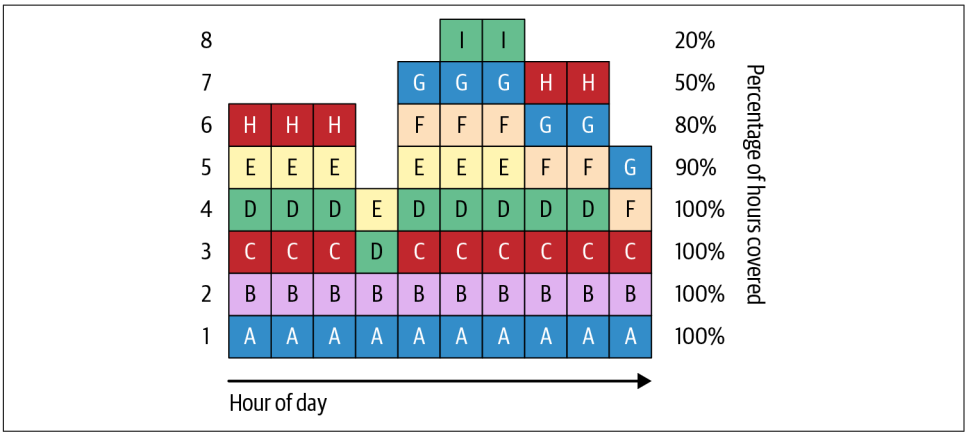


Figure 18-2. Commitment utilization

If you purchased four units of commitments, they would each have 100% utilization; a fifth commitment would have 90% utilization; a sixth commit, 80%; and so on. If the commitment saves you 25%, then, to save money compared to the on-demand cost, you would need to be utilizing the commitment more than 75% of the time.

Knowing that, committing to seven commitments, you would be paying more for that seventh commit than the on-demand rate, so any usage above six commitments isn't coverable.

In the rows above line 2, a single commitment may be applied to different resources. Further, these resources may come from different accounts within your cloud estate. It's this sharing of resources—and the economy of scale that comes with it—that makes centralized commitment buying a core tenet of FinOps best practices. If you drive commitment purchasing solely from the team level, you may end up with the wrong number of commitments. We've seen that it's best to divorce rate optimization from the daily lives of teams so that they can focus on optimizing their resource utilization.

Because this one waterline chart applies only to things that might be covered by a commitment we're considering, and because resource usage might be coverable by a variety of commitments (an SRI, a Compute SP, an EC2 SP, etc.), our process (and our automated recommendation tools in most cases) will be looking at many variations of consistent usage and the coverage rates that will result in the most savings for us, and presenting the FinOps team with options it will have to consider very carefully. It is rare to do this sort of waterline analysis by hand anymore, but

knowing how this analysis is being done is important so that you can balance the different recommendations you receive from your tools, vendors, and teams.

The new role of procurement is to be a strategic sourcing department that ensures your company is getting the best possible rates on resources consumed, via commitments and negotiated pricing.

Step 2: Understand Your Level of Commitment to Your Cloud Service Provider

No matter which cloud service provider you use, the goal for commitments is to save as much as possible. Before you develop a commitment strategy, there are a few things to consider. Your company's profile will affect the strategy of your commitments. You also need to consider the level of overall commitment you have to your cloud service provider, negotiated rates you may need to factor into your purchase decisions, payment processes, impacts to cost allocation, and the overall availability of capital for any up-front payments.

When you create a commitment with a cloud service provider, there may be an up-front charge and an hourly charge. When you combine all charges over the full term, you get to the total commitment cost—that is, what you are committing to spend with your cloud service provider.

All three of the big cloud service providers allow you to commit to either one- or three-year commitments, with the longer commitment giving you better savings. To decide between the two options, you need to consider your commitment to your cloud service provider. What is the likelihood of you reducing your usage during this term?

Let's say that you forecast that you are very unlikely to move off your current cloud service provider within the next 12 months, but you are less confident about a commitment for the next three years. In that case, you shouldn't commit to a three-year commitment-based discount program. The extra savings you receive won't be realized if you stop before the full three-year term.

Step 3: Build a Repeatable Commitment-Based Discount Process

The second step in building your commitment strategy is to build a process that you can run regularly and repeatedly to make the right buying decisions as quickly as possible. Why is this important? Because cloud is all about just-in-time purchasing. You spin up resources just when you need them. Likewise, you should be buying commitments just when you need them: not too early and not too late.

The business approval process can also contribute to commitments being made on an infrequent schedule. If the business process takes many weeks to get approval, then performing commitments monthly would be troublesome. This is where FinOps can benefit the business by building out new processes for performing commitments, with the understanding that improving the process saves the business even more.

To do this, you need to know who is going to be involved with the process, what decision criteria you need to consider before you commit, and what the sign-off process looks like. Consider having this coded into a company policy, outlining who is responsible, how approvals work, and, ideally, a preapproved process for set amounts of commitment purchases. Set boundaries on the policy for it to be reassessed after major events like new product announcements—such as new SP offerings—that impact the decision process of making commitments.

We've seen successful companies allocate a quarterly budget to the FinOps team for commitments, with additional approvals needed only if the overall commitment is over this budget. Using forecasts to predict expected additional commitments needed and existing commitment expiries, you should be able to provide your finance teams with an expected spend on commitments in the next 12 months. In most cases, reducing the number of people and the amount of paperwork involved in the approval process speeds up the commitment frequency. We'll talk more about metric-driven cost optimization in Chapter 22, where we'll see how having a preapproved spend amount can lead to automated commitment purchases.

Your infrastructure in the cloud should be fluid: rapidly changing instance types, migrations, elasticity, usage spikes, and so on. You should support these changes by rapidly adapting commitments to match. You may need to make monthly purchases to keep up with the rate of change, and you will almost certainly make more frequent exchanges, modifications, and adjustments to your portfolio.

Thus, your FinOps team owns the commitment process. They need to have a deep understanding of how commitments work (see Chapter 17) and a sensitivity to the motivations and future plans of the teams running the resources against which they are making commitments (see Chapter 3). They will need to interface with finance, tech, and executive teams to carry commitments over the line.

Why should you centrally manage commitments? The easy answer is that central management can save your company 10%–40% on your cloud bill. The more nuanced answer is that the job of deciding when and how many commitments to buy is less efficient when every individual team decides what to buy. It is very rare that any team uses every resource consistently enough, every hour of the day for one or three years, to buy coverage for all of their resources. Spiky or variable usage that's not running consistently enough will be uncovered and cost the company on-demand rates. If you add up that little bit of on-demand slack from every team in your organization, it results in 10%–40% less efficiency.

Additionally, commitment purchasing is typically not done well by tech or ops teams, who are focused on running their resources efficiently. And you don't necessarily want them to spend their valuable time keeping up with the ins and outs of commitment-based discounting programs from your cloud providers. Remember that you want to create a division of duties, where the FinOps team performs *rate optimization* while the line of business teams perform *usage optimization*.

The central FinOps function has a view to all usage, across the organization, and can do the best job of maximizing coverage by looking not just at the waterline usage of one team but of all teams. As you learned in the previous chapter, commitment benefits can be made to flow between accounts, subscriptions, or projects, so you want those with the best visibility to the organization's overall spend to be in charge.

If you have a decision-making strategy for purchasing your commitments, and your central FinOps team is making these purchases based on the waterline of usage and planned future use of resources it gets from your engineering or line of business teams, then your next decision is often where to buy or attach the commitments that you purchase.

You learned in Chapter 17 that you can purchase commitments at various levels in your account hierarchy, so that they apply to resources in different ways. As an example, if you are running an AWS management account, with three member accounts linked to it, you have the option of buying the commitment in the management account, or in one of the member accounts.

If you assume that you have no significant resources in your management account (a good policy to be sure), then buying a commitment inside that account would immediately have it flow down to the other three accounts' resources, applying to the smallest (in the case of ISF-based RIs) or highest discount (in the case of SPs) resources in those other accounts, randomly.

Another option would be to purchase the commitment within one of those member accounts where there are a substantial number of resources. Because of the way coverage is applied, AWS RIs or SPs would apply to *all* eligible resources inside that account first, and then begin to apply to other resources in other accounts. You might wish to do this if, for example, you had one production account, expected its resources to be most consistently running, and wanted all resources in that production account covered first, you might purchase the commitment in that account. Or you might have one team that has paid for the up-front fee on the commitment and wants its resources covered first.

One note to consider here relates specifically to SPs. Because SPs uniquely apply to different types of services (EC2, Fargate, Lambda) and the discounts are significantly different for those services (up to 75% for EC2, 17% for Lambda), you should be careful to note that if you purchase an SP in a member account that has both EC2 and

Lambda resources, that commitment will cover *all* eligible usage in that account first. So you may find that your SPs are covering substantial amounts of Lambda serverless usage before they are flowing to other accounts where the savings would be at a much higher percentage (and subsequently more monetary savings).

So your strategy for where to purchase your commitments should consider this.

One approach we have seen among some more advanced customers and in use by some vendors who offer commitment management *as a service* is the use of so-called *sidecar* accounts: dedicated member accounts which *only* contain commitments, no usage at all. Buying commitments in a specially designated account for them eliminates the need for the FinOps team to access the management account (again, always a good thing to limit access here), and gives customers the ability to potentially move the commitments around between organizational units to have them apply differently over time.

This can be particularly valuable as an approach for managed service providers or companies that routinely buy or sell parts of the business through merger or acquisition. If commitment-based discounts are attached to an account that is operating a system that is being sold, the sidecar account with the commitment can simply be repositioned in the organization as those accounts are moved to a new management account in its new home. Had the commitments been purchased in those accounts, they would have moved with the resources.

Commitments might all be purchased in one sidecar account or in several to help influence where coverage might apply. Not every organization needs to obtain this level of control, but if you do, this can be an important aspect of your overall commitment strategy.

Step 4: Purchase Regularly and Often

Cloud is designed for just-in-time purchasing. You shouldn't be running infrastructure or purchasing capacity (i.e., buying commitments) in a data center fashion.

Years ago, AWS had graphs like those in Figure 18-3 on their home page. The graph on the left showed the way you had to buy capacity in the data center and hardware world, fraught with constraints and long lead times for hardware. The graph on the right shows the ideal way to run infrastructure in the cloud: you spin it up just when you need it. The same is true for purchasing commitments.

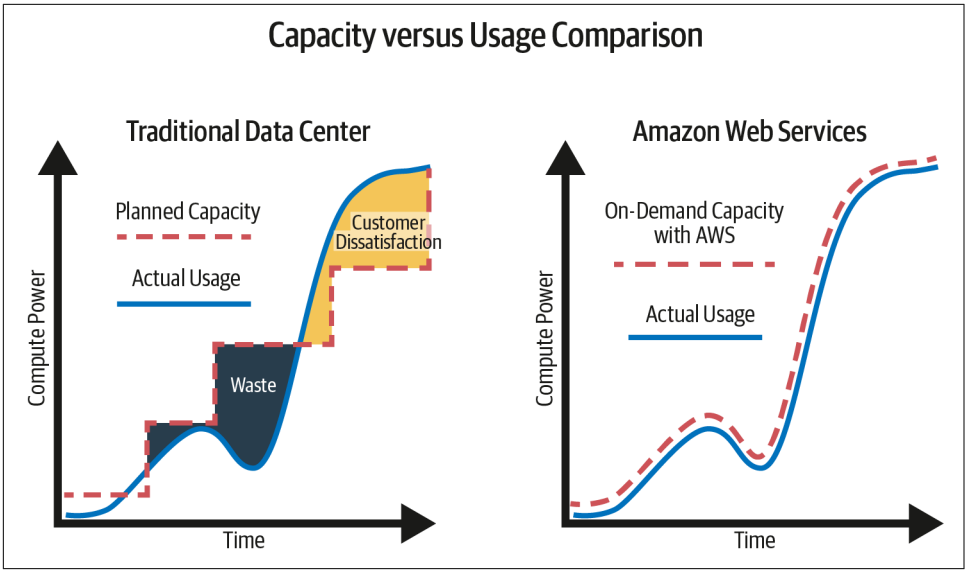


Figure 18-3. Diagram AWS used to host on their website

Step 5: Measure and Iterate

We have heard many horror stories of companies that have purchased commitments incorrectly. To reduce the risk of this common mistake, make sure your initial purchases are small. You should have carefully set metrics with thresholds. These can start with simple measures, such as what percentage of cloud spend that should be covered with a discount *are* covered with a discount program. Conversely, you should track the amount of waste (or vacancy) of existing commitments to ensure they are used to their maximum extent. For more on the metrics to use in managing commitments, we cover metric-driven cost optimization in Chapter 22.

One frequent cause of overpurchasing of commitments or purchasing incorrectly is the tools that recommend what types of commitments to buy! Commitment recommendations, like any of the automated processes performed by FinOps tools and platforms (from the cloud providers or from third parties), are primarily based on expert analysis of your past usage. To the extent your future usage looks exactly the same as your past usage (unlikely) these forecasts can be very accurate. However, you have the power to look back at a variable time period to generate recommendations.

One contributor told us the story of purchasing a large quantity of AWS RIs to cover a large amount of on-demand cost. The purchase began to make a measurable difference immediately and the business was supportive of increasing the coverage by making another incremental purchase of RIs only two weeks later. However, the tool being used to generate recommendations had a look-back period of 30 days, so it was generating recommendations based on two weeks with that new RI coverage and two

weeks without it, which caused it to recommend far too many RI purchases for the second go-round.

Purchasing commitments is about managing risk; the more commitments you make, the more risk you carry of having them become underutilized. The better your forecasting capability—see Chapter 13—becomes, the more comfortable you will be at making larger commitments safely.

There is a lot to take into account as you purchase commitments, so take your time to analyze the impact of your purchases, and be sure of each incremental step toward higher coverage. As your experience and confidence in your process grow, you can increase both the frequency and the size of your purchases.

Step 6: Allocate Up-Front Commitment Costs Appropriately

Chapter 4 introduced the matching principle, which holds that expenses should be recorded in the period in which the value was received. In AWS, when a commitment is paid up front, even partially, it applies a discount to your usage over a long period (one or three years), and you need to distribute the up-front payment over the term. Distributing the up-front payment is called *amortization*. With AWS, you are given the option of paying all, some, or nothing up front. You receive a higher discount for the commitment the more you pay in advance. With Azure you have the option of paying up front or monthly, with no difference in price. And with Google Cloud, you pay monthly, so there is no amortization to track.

Deciding whether or not to pay up front requires two major considerations:

Accounting
: The first is the impact to your accounting processes. Chapter 17 highlighted that commitment discounts will apply randomly across your cloud usage. If you need to allocate part of the up-front payment to one team and another part to another team, you will be required to track exactly where the commitment applied its discounts and then amortize accordingly.

Granularity
: You can't just divide a commitment into equal monthly parts. Not all commitments start on the first hour of the first day of the month, and February is 10% shorter than March, so the amortization requires a more granular approach.

We will cover who pays for commitments later in the chapter.

While commitments save money, they can require available cash to apply to up-front fees. For some organizations, this isn't an issue, while for others it is prohibitive to business activities. Some organizations need to spend money inside particular time periods, such as when a budget must be consumed before the end of the fiscal year.

Where the money is available, we see experienced FinOps practitioners considering the *weighted average cost of capital* (WACC) and the *net present value* (NPV) of cash. These calculations determine whether applying the business capital to up-front commitments is a good investment versus other possible investments. While there's a lower discount for not paying up front, investing some money somewhere else in the business might have better financial outcomes.

Think of the premium charged by AWS for each of the *No upfront* and *Partial upfront* purchase options as being equivalent to an interest charge for the vendor having lent the non-up-front portion of the purchase, and think of that interest charge as carrying an annual interest rate. We refer to this as the *implied interest rate* of the *No upfront* or *Partial upfront* purchasing options.

Whether a consumer of cloud resources should choose the *All upfront* or alternative payment options depends on whether the consumer's WACC is lower or higher than the implied interest rate charged by the service provider to defer payment of the commitments.

A consumer's WACC (sometimes called the *internal cost of capital*) is the cost of financial capital to the cloud consumer in the form of interest costs for the company's debt, and the cost of the company's equity capital. The process of determining which cost is greater involves calculating the NPV of the purchase cash flows for each of the *All upfront* and *No upfront* scenarios, respectively.

The process begins by determining the consumer's WACC. Most enterprises will have a standard WACC in use by the finance department for decisions of this type, and in general you can take this figure as a given.

If the discount rate calculated—using techniques like Excel's Goal Seek—exceeds the client's WACC, the client should purchase *All upfront* commitments. If the figure is lower than the consumer's WACC, then the implied interest rate of the *No upfront* commitment purchase option will be less expensive to the consumer than the AWS's *All upfront* option.

The mix of commitments purchased changes this imputed interest rate for each purchase, so this can be done for each purchase decision, or it can be done for representative purchases to help a customer set a policy of buying no, partial, or all up-front RIs or SPs, and only revisited occasionally. Note this applies only to AWS commitments as they are the only ones that have a variable discount when paying up front.

How to Manage the Commitment Strategy

To reduce the demand for individual teams to manage their own commitments, it's important for the FinOps team to provide visibility into what discounts are applying to each team's resources, along with the overall savings achieved. Without this visibility, teams will be tempted to commit their own usage in order to gain savings. Having distributed teams buy commitments when you're operating in a centralized commitment-based discounts model is dangerous, as you're likely to end up overcommitted.

A centralized commitment-based discount model doesn't necessarily prevent individual teams from having input. The FinOps team should capture the roadmap items that operations teams are planning to implement, especially where these changes will impact usage on the cloud. When a commitment can be converted or exchanged, less input from individual teams is required, as the FinOps team can convert the commitments when needed. If a commitment cannot be modified, then the individual teams who own the resource usage being covered should be given the chance to comment on the commitment being made. Making the responsible teams aware of the commitment and of what that means for their usage ensures that they'll take the commitment into account when planning any change to the resource usage in the future.

Purchasing Commitments Just-in-Time

We've heard many opinions on how companies should time their purchases of commitment-based discounts. It's important to understand the impacts of committing too early versus committing too late. If you commit earlier than needed, you'll be paying for the commitment without any discounts being applied to your account. If you commit to more usage than you end up needing, you will be paying for the commitment for potentially the full term (one or three years) without any savings at all. If, however, you wait too long, you'll be paying the on-demand rate for your usage and will miss out on the savings you could have otherwise been receiving had you committed earlier.

To help you understand this better, Figure 18-4 shows some resource usage in a cloud bill. If you purchase a one-year commitment in January, you will end up paying for the commitment for three months without obtaining discounts, wasting 25% of the commitment (3 months / 12 months). When the usage starts in April, you will immediately receive the discounted rate for this usage until the end of the year. If this commitment doesn't save you more than 25%, then you will end the year paying more than you would've had you not made the commitment at all.

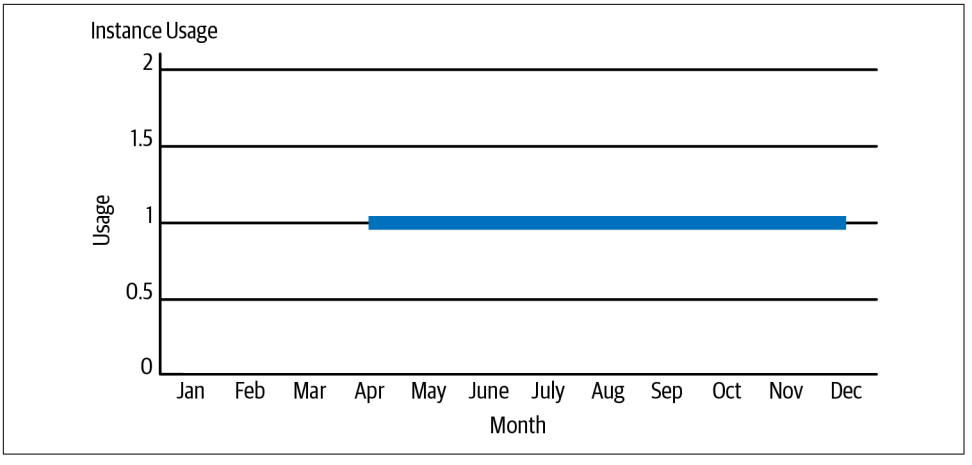


Figure 18-4. Resource usage starting in April and running for nine months

Inversely, if you had waited until September to make a commitment, you would end up paying for five months at the full on-demand rate, missing out on savings you could have realized.

We will cover a more mature model of performing commitments only when needed in Chapter 22. Most companies start out making their commitments on a set schedule, such as yearly, quarterly, or monthly. When there's a set schedule, companies are tempted to overcommit to achieve the most savings during each cycle.

For example, a company that makes commitments yearly might commit to 30% more commitments than they need. The idea is that as the year progresses they have an increased cloud spend, and the spare commitments will start applying a discount immediately. However, during the first part of the year, the company is overpaying for these extra commitments that are not applying a discount. As shown in the previous example, there is a risk that these commitments never apply more discount than the amount of unused commitment costs.

By increasing the frequency at which you purchase your commitments, you reduce the amount of time you're charged at the full on-demand rate, and you also remove the temptation to overcommit on your commitments. There are a couple of roadblocks to performing commitments at a higher pace, however. The time it takes you to analyze the amount of commitments you have and need can slow down the frequency of performing commitments. If your cloud service provider has built-in services that make recommendations on what to commit to, this is a good place to start. As your FinOps team matures, using both third-party tooling and analytics you build yourself will help improve the quality of your recommendations and speed up the analytic process.

When to Rightsize Versus Commit

You saw how rightsizing impacts commitment-based discount programs in Chapter 15. For this reason, many people believe that the best approach is to rightsize first and commit later. As correct as this approach sounds, it is not always the right one. It invariably leads to lost savings opportunities through unnecessary on-demand spending.

Usage optimization and rightsizing are hard and can take a long time to both burn down debt and begin to do it proactively. It requires engineering teams to get involved, be educated about cost, and buy in to making changes. It doesn't happen overnight. Often, teams are focused on delivering features and cannot jump right into usage optimization. It often takes six months or more to change to a culture in which engineers consider cost as an efficiency metric. In the meantime, if you don't commit anything, you're needlessly paying on-demand for a very long time. And, when rightsizing does happen, it's usually a gradual process rather than a big bang, as engineers start baking optimization into their ongoing processes.

So look at both rightsizing and commitment as ways to optimize your overall spend. These concepts seem to be in opposition, but only in totality. Doing a little rightsizing at a time on the items where the biggest impact can be made (and you know you have available engineering resources to get the job done timely), and doing a little commitment purchasing to cover the most consistently used resources, and then repeating that again and again creates a ratcheting effect that drives the business incrementally toward better cost optimization.

One of the biggest mistakes companies make when their FinOps practices are immature is waiting until they have *completed* optimizing usage before they commit anything. Remember, you progress through the FinOps lifecycle many times, each time making the most critical optimizations. We shouldn't be debating when to rightsize versus commit. It's nearly always a balance between rate optimizations and usage optimizations driven by the company's goals.

While we do advocate for centrally managed commitment planning, optimization of the right resources should be decentralized and constantly improved upon. The most effective FinOps practices put more effort into making sure things are optimized in the first place, rather than reactively trying to fix them later. Shift cost into your engineering processes. Think about the size of the resource up front. No matter how good you are at that, however, there will always be times in a workload's lifecycle where a resource is oversized. It might be right today, but five deployments later you become more efficient. You upgrade the library and now it uses half the memory.

There is no "We never have to rightsize; we get it right in the first place." It's iteration, evolution, etc.

In the early stages of usage reduction you may be getting rid of unattached storage volumes, or turning off 100% idle machines in development or staging accounts. Early stages of rate reduction may be buying your first 10% of coverage (maybe even with *No upfront* RIs if you're using AWS). Gaining maturity and complexity in each will require cross-functional, cross-team collaboration and a lot of education.

The best path is to set conservative goals for both and to start parallel processes from the beginning. It is unlikely that a large proportion of your entire cloud compute spend is being wasted and could thus be resized. Therefore, you could likely cover 50% of your infrastructure safely with commitment-based discounts, especially if your cloud usage is growing over time due to migrations or new applications.

When starting out, you should set a small commitment coverage goal, maybe 25%–30%, and commit some portion of your cloud resources, thereby realizing savings early. As you reduce your waste, your newly purchased commitments will apply to a greater percentage of your remaining usage. Keep in mind that programs like instance size flexibility (ISF), AWS's Convertible RIs, and Azure's ability to cancel RIs give you room to optimize your infrastructure usage, even after you've committed.

Since you're able to measure the amount of unused commitments in your accounts, you can reassess your coverage goal and ideally increase it over time. A maturing FinOps practice will see you getting on a regular cadence of buying commitments, while the engineering teams regularly clean up unused resources and rightsize underused ones. An advanced practice will get into metric-driven cost optimization (Chapter 22).

The Zone Approach

Many commitment-based discount programs don't apply globally, meaning they apply only to the same region (and, in some cases, the same availability zone). Also, they apply only to resources matching the same attributes, as detailed in Chapter 17. With this in mind, commitment-based discounts aren't often purchased in low-use areas. Commitments in a region with low use has a risk of being uncovered, therefore costing the business instead of saving. In the same vein, commitment-based discounts for older-generation instance types have similar risks.

When teams build infrastructure in low-use regions and/or use instance types that are deemed too risky to commit to, they won't gain any discounts. Dave Van Hoven, Senior DevOps Manager at HERE Technologies, has an approach to building visibility into these decisions of not covering specific usage types and/or locations, which allows engineers to make informed decisions about how they build.

The *zone approach* is to publish information internally to the business on locations and instance types that will be automatically factored into commitment purchases by the centralized FinOps team (the allowed zone), and locations and instance types that won't be covered by commitments without asking the FinOps team (the restricted zone).

The allowed zone
: The allowed zone is the easiest zone for engineers. Choosing to build using more recent and common instance types, and then deploying into business-supported regions that are more broadly adopted throughout the company, enables engineers to ignore commitment-based discount programs in their day-to-day considerations. This allowed zone helps drive engineering decisions into areas that the business wants most to support.

The restricted zone
: The restricted zone requires engineers to either manage their commitments themselves or work closely with FinOps to purchase specialty commitments. Any changes in the engineering team's infrastructure will always require them to consider the extra commitments they have made, as it will be unlikely that the commitments can be modified to be used by another team. The restricted zone requires extra effort, and in turn becomes less attractive for engineering teams.

The blocked zone
: A small extension of this concept is the blocked zone, which clearly states which locations and types of commitments the organization does *not* approve of purchasing. This can be used to delineate where engineers shouldn't be building infrastructure.

The zone approach makes it clearer how a centralized commitment practice works. It shows teams where commitments will not automatically apply to them and defines a process to apply commitments to obscure usage. Reporting should be provided to highlight the efficiency and savings being generated not only by the centralized FinOps commitment process (in the allowed zone) but also by the restricted zone.

Using more flexible commitment options like SPs or Flexible CUDs to cover resources in less-used areas allows the FinOps team to cover miscellaneous use in regions that don't get regular RI coverage. This mixing of commitment types is much like purchasing different types of investments to hedge against exposure risks of various sorts. In this case, the risk of teams moving away from RI coverage in less used regions can be lessened by having more flexible (though less discounted) coverage that applies in any region.

Who Pays for Commitments?

Approving a FinOps team to *purchase* commitment-based discounts doesn't necessarily mean they will *pay* for the commitment. There are two main components of the cost of a commitment: any up-front fee and the hourly fee. Some commitment-based discounts consist of only one of the two: *All upfront* and *No upfront* commitments.

With any up-front payment, the cost should be *amortized*, that is, applied to the cost of the resource it ends up providing its discount to each hour, since the commitment is applied to your cloud billing. Any unused capacity of the commitment (which in FinOps we call *vacancy*) is then charged against your bill directly and left unamortized from the initial up-front fee. This vacancy cost is usually found in the account, subscription, or project where you purchased the commitment.

Once the cost is amortized correctly at the end of the month, you'll end up with:

- Prepaid expense accrual of the remaining up-front fee to be amortized in future months
- Amortized cloud resource costs incurred in that month (cost allocated via tags, labels, and accounts)
- Unamortized costs of unused commitment charged centrally in the billing data

When a FinOps team is approved to spend on commitment purchases, they add money to the prepaid expense accrual. This isn't a business expense as far as monthly costs go. In accounting, when using the matching principle of the generally accepted accounting principles, the business prepaid expense is recorded as an asset, and it will be turned into an expense when you amortize the cost into your cloud costs. Thus, the FinOps team does not report the budget for commitment purchases as a business expense.

Check with your accounting team on how they handle the prepaid expenses for commitments and align reporting to match this.

When costs are amortized out to your cloud resources, you can include that amortization in showback reporting using tags, labels, and accounts, according to your cost allocation strategy. You can attribute the commitment costs to your showback/chargeback cost allocation process and handle them as you do all of your other cloud costs.

For the unused commitments, however, charges are centrally billed (to the account they are purchased in) and will need a process to allocate them appropriately. Here are four strategies we have seen used to handle this unused commitment cost:

Account-based attribution
: If commitments are purchased inside team-owned accounts—close to the usage to which they are intended to apply discounts—it's possible to distribute the unused commitment costs, based on the account in which the commitment is purchased. The charges are allocated to each team owning the account.

Blended amortization
: Adding the unused commitment cost to the amortization rate effectively blends the unused cost into the rate everyone pays for resources covered by that commitment. We aren't referring to the AWS billing data concept of a *blended rate*, which we discussed in Chapter 4. This leaves uncovered resources at the normal on-demand rate. The variability of which exact resources a commitment applies to means that sometimes a resource will get a discount applied, while at other times it won't. This variance in discount allocation can make tracking team costs more difficult to understand because there can be variance introduced to the periodic cost due to rate optimization deltas the team itself had no way to control. Management and finance awareness of this is paramount so that these groups do not unjustifiably criticize engineering teams for this variability caused only by the way the cloud provider applies commitment coverage. We recommend using resource-blended rates when performing showback/chargeback because, despite this variability, it reflects how actual discounts were applied, and it will match cloud-provider tool reporting of costs.

Resource-blended rate
: Here you create a new rate for commitment by blending the total costs of resource rates across the bill. Once again, this is different from the AWS blended rate you see in the AWS billing data. Instead, this concept takes all usage matching a commitment (on-demand and discounted), applies the up-front amortization and unused commitment amounts, and then creates a new rate across all of the usage. This means everyone using that type of usage will receive the same rate, whether or not a commitment was applied in the billing data. This essentially creates a company's own averaged rate—in between the on-demand and the discounted price—for each resource type that changes as commitment coverage does. It avoids having *random winners and losers* wherever a commitment applies. However, it is a significant amount of work to produce this model of reporting in a way that is understandable to all, and it creates small differences between blended rate reporting and any cost information your users will see in cloud provider native tooling.

Savings pool
: Build out a process with your finance team that centralizes savings, with all reporting, budgets, and forecasts using on-demand rates. Engineering teams aren't required to think about the correct CUD rates for their usage, and the unpredictable application of commitments to usage is no longer an issue. All

savings are reported centrally to the FinOps team, from which the unused commitment costs are subtracted. This savings amount works as a reverse budget, where a set amount of savings is expected by the FinOps team, and performance is measured based on how close they come to achieving the correct savings amount. This central savings pool is then distributed to the business in a more controlled and appropriate manner. This approach works well as it can eliminate the variability of how coverage is applied, and it allows engineering teams to focus entirely on usage optimization. However, it means the engineers are looking at on-demand costs, and that finance and product teams may need to look at the blended amortization cost metrics to correctly calculate actual product margins. This could also hide important aspects of spending that might allow engineering teams to focus more effort on optimizing resources that either aren't or can't be covered by commitments, as they can't see what is being covered and discounted.

The approach you take to assigning costs of commitments will depend on your business processes and possibly will change over time as your FinOps practice matures. As with many aspects of cloud cost management, there are a variety of ways to look at cost, some of which will seem unfair, inequitable, too variable, or downright punitive to some teams and persona stakeholders. But by creating approaches that are transparently representing your costs, and by having all persona stakeholders involved in how your distribution of costs is structured, you can set expectations such that even if something seems unfair to one team, everyone recognizes that it seems that way and can compensate or account for this in their responses.

Strategy Tips

You might find the following pointers useful when building out your commitment-based discount strategy:

Build visibility first
: A story we've heard time and again goes something like this: "I purchased a commitment, but my bill just went up." Often commitments are purchased to stem cloud cost growth, and they are first implemented in the growth phase of cloud within an organization. Most likely, cloud growth is larger than the savings the commitment is able to make, and without any visibility into how much a commitment is saving, all you have is an increasing amount of cloud spend.

Understand how commitments work
: From Chapter 17 you should have gained an appreciation of just how complex commitment-based discounts programs are. The most common mistake people make when buying commitments is not understanding them correctly and then buying the incorrect type of commitment, or buying them in the wrong

locations. When a commitment doesn't match the usage you are expecting, it can lead to very large cost increases.

Get help
: FinOps practitioners don't prove themselves by doing everything alone. The most successful practitioners in the FinOps Foundation use some form of outside assistance. The FinOps Foundation is itself an outside resource that is there to assist you.

  By learning from your peers, attending training courses, getting certifications, and engaging with your cloud service provider's support channels, you can ensure that your commitment-based discount program will be most likely to succeed. If commitments are new to your organization, you should consider using an experienced consultant to get started.

  Remember that if you are paying for support from your cloud vendor, you also are paying for time for your Technical Account Manager or TAM. Some customers never proactively ask for help from their TAM, but commitment purchases are a great thing to have TAMs validate. In fact, there is value in getting everyone who can to validate commitment purchases, as the more eyes the better, and having to justify your decisions to everyone before the purchase can be clarifying. Nearly every practitioner we know has a story about buying commitments incorrectly, so ask for help in this area in particular.

Avoid going from zero to hero
: When you are earlier on in the maturity of your practice, we recommend buying just one commitment. This allows you to confirm your knowledge with a low-risk purchase. You can verify that you've avoided all of the common mistakes before you make a more substantial purchase. Trying to go from "zero to hero" often leads to failure. You can consider the zone approach we discussed earlier, identify the lowest-risk usage that could benefit from commitments, and then purchase in increasing amounts. It's a great way to build confidence in your strategy, and it staggers the renewal of commitments in ways that allow you to adjust your commitments over time more incrementally too as they expire.

  But you shouldn't delay. While we recommend starting slowly, not starting at all is a major—and common—mistake. Trying to avoid mistakes by not making any purchases leads to continued overspending. Once again, find and purchase some low-risk commitments and then continue learning and growing your reporting processes as you start to see those savings.

  When you're just starting, we recommend setting commitment goals low. Consider 25%–30% commitment coverage in the allowed zone. As confidence grows, revise the goals upward. A common coverage amount we see organizations aspire to is 90% commitment coverage.

Consider the overall impact
: A commitment is going to apply somewhat randomly to a cloud bill. So if you have existing cost reporting and teams tracking spend closely, a commitment purchase can cause confusion when spending drops in one area but not another. Communicating the purchase of commitments, and iterating on reports used by the business to accurately reflect the impacts of the savings they generate, is key to avoiding confusion.

These tips are the most important ones we believe you need to keep in mind; however, for the most support, connect to your peers in the industry, use experienced contractors, and utilize FinOps platforms to guide your first steps into commitment-based discount purchases.

Conclusion

FinOps strategies for managing commitments vary from organization to organization. But there are a few common practices that lead to efficient and well-managed commitment fleets. Having centralized management by the FinOps team is one of the most important.

To summarize:

- You should use all the help available in building your strategy, and avoid building it alone.
- You must learn how commitments work and spend time building visibility before your commitment purchases.
- You shouldn't delay buying commitments. Purchase low-risk commitments first, and then build on your processes to drive up coverage.
- Communication is important to your entire organization to avoid confusion as to who is doing what, and how to accurately look at your costs.
- You will likely use a variety of types and configurations of commitments to achieve your overall commitment goals.
- Start small and increase your impact incrementally as you gain maturity, gain confidence in your strategy, and see the impact of each commitment you buy.

A correctly implemented commitment-based discount strategy will evolve and mature with FinOps, saving an organization large amounts. Ideally, metrics are used in all FinOps processes.

In the next chapter, we will look at sustainability and how FinOps is enabling the growth of a new emerging practice called *GreenOps*.

CHAPTER 19

Sustainability: FinOps Partnering with GreenOps

Over the past couple of years, there has been a big increase in the focus and awareness on sustainability as it relates to cloud consumption, with cloud providers and cloud teams everywhere suddenly talking about measuring and reducing their cloud carbon footprint.

Sustainability as a discipline has three major pillars: environmental, social, and economic. The focus in this chapter will be on the environmental pillar as it relates to emissions from cloud data centers.

We will cover why this matters to a FinOps practitioner later in this chapter, but first consider why your organization should care. The deeper focus organizations are putting on the carbon footprint of their digital services is driven by pressures coming from a number of important internal and external stakeholders:

- Investors and shareholders are pushing CEOs and board members to hold them accountable for their overall emissions, driven by ESG (environmental, social, and governance) initiatives and the impact of publicly reported measures of sustainability.
- Governments are tightening environmental reporting requirements and regulations, so there is less ability for *greenwashing* (falsely claiming or promoting that a company is more environmentally friendly than it actually is) and more focus on action.
- Customers are increasingly making buying decisions on ethical grounds including the sustainability practices of the companies they are considering.

- Employees are demanding their own companies to be more sustainable. Employees are placing a huge importance on how prospective employers are addressing sustainability when selecting their next place of employment.

These pressures combined have successfully raised the profile of sustainability inside the boardroom, creating an increasing focus on sustainability targets and/or net-zero deadlines, with an additional focus on accountability, transparency, and delivery of measurable improvements.

> *Nothing motivates the C-suite more than something that affects their bonuses. This means that critical things like budget and time are being made available to focus on sustainability.*
>
> —Mark Butcher of Posetiv Cloud on the FinOpsPod podcast Episode 8: FinOpsPod Voicemail: GreenOps

So why does sustainability matter inside the world of cloud? Because of the sheer size of the carbon footprint generated by cloud services. Until relatively recently, the carbon footprint of IT services has been ignored by ESG professionals, with their focus being elsewhere in the business, with IT and cloud consumption in particular being seen as irrelevant when compared to other, more directly polluting, areas of the business.

But the industry and the media have become more vocal about the carbon footprint of data centers with the growth of cloud services, with a focus on how much electricity and water is being consumed to feed the seemingly insatiable demand for data storage and compute. It makes sense that this shift is happening now as more and more companies are closing their own small data centers and consolidating to the cloud. The aggregation of workloads from many small facilities into the massive complexes run by cloud providers allows for even small increases in overall efficiency enacted by a cloud provider to have a much more significant impact, in the same way that addressing the emissions of utility-scale electric generating plants has a bigger impact than making changes at individual factory power plants or backup generators might.

The data centers that comprise the cloud reportedly consume as much as 1%–4% of total global energy, with this figure expected to grow as high as 10% within a decade, dwarfing other market sectors such as aviation. Further, they are massive consumers of water, which creates emissions in the form of water vapor. "The typical data center uses about 3–5 million gallons of water per day, the same amount of water as a city of 30,000–50,000 people," said Venkatesh Uddameri, professor and director of the Water Resources Center at Texas Tech University in a 2021 NBC News interview.

According to a 2022 *Time* article, "Google considers its water use a proprietary trade secret and bars even public officials from disclosing the company's consumption. But information has leaked out, sometimes through legal battles with local utilities and conservation groups. In 2019 alone, Google requested, or was granted, more than 2.3

billion gallons of water for data centers in three different states, according to public records posted online and legal filing."

As companies move away from their own data centers, driving an explosion of cloud consumption, sustainability measures need to follow.

What Are Cloud Carbon Emissions?

To manage and reduce your digital carbon footprint, everyone involved needs a common understanding of what carbon emissions are and how they are measured. There are many domain-specific terms used within sustainability, most of which we will not cover in this book. If you are looking for a list of these terms and an explanation of their meaning, you can find one on the FinOps Foundation website. Importantly, for this chapter you need to understand what greenhouse gases are and the unit of measurement often found in carbon reports:

Greenhouse gases
: Greenhouse gases are gases in the atmosphere that absorb and reemit heat. Generally, the more of these gases in the atmosphere, the hotter the earth will become. Greenhouse gases include carbon dioxide (CO_2), water vapor, methane (CH_4), nitrous oxide (N_2O), ozone, and more.

CO_2e
: Often confused with the notation of carbon dioxide (CO_2), CO_2e is the *carbon dioxide equivalent*, which includes all greenhouse gases as one normalized metric. The global warming potential of each gas is used to convert the measurement to the equivalent amount of CO_2 impact on the environment. CO_2e gives you one metric to represent the impact of all the gases combined.

  You may find *CO_2e* written in many ways, such as: *CO_2eq*, *CO_2-e*, *$CO_2equivalent$*, or *CDE*.

Net-zero
: Used as a way of describing an outcome where the total emissions from an organizations' activities have been offset via programs that reduce carbon emissions. Emissions should be reduced by the organization, measured, and then offset. You can read more about net-zero at the Science Based Targets website.

The total carbon emissions that an organization is responsible for are broken down into three categories, known as *scope 1, 2, and 3*. The three scopes are a way of grouping the different kinds of emissions a company creates in its own operations and in its wider *value chain* (i.e., suppliers and customers). The name comes from the Greenhouse Gas Protocol, which is the world's most widely used greenhouse gas accounting standard.

Developing a full greenhouse gas emissions inventory covering all scopes enables companies to understand their full value chain emissions and focus their efforts on the greatest reduction opportunities. Doing this is really important, as the IT industry has been somewhat guilty of focusing only on power as the primary source of carbon emissions, ignoring the bigger picture of emissions from all three scopes.

Scope 1, 2, and 3 Emissions

Essentially, scope 1 and 2 are those emissions that are owned or controlled by a company, whereas scope 3 emissions are a consequence of the activities of the company but occur from sources not owned or controlled by it.

Scope 1 emissions
: Emissions from sources that an organization owns or controls directly—for example, from burning diesel in on-premises data center generators.

Scope 2 emissions
: Emissions that an organization causes indirectly when the energy it purchases and uses is produced. For example, for an on-premises data center, the emissions from the generation of the electricity powering the data center would fall into this category.

Scope 3 emissions
: Emissions that are not produced by the company itself, and not the result of activities from assets owned or controlled by them, but by those that it's indirectly responsible for, upstream and downstream emissions from the companies value chain. An example of this is when you buy, use, and dispose of products from suppliers. Scope 3 emissions include all sources not within the scope 1 and 2 boundaries. Scope 3 is where 100% of your cloud provider emissions are categorized.

It is easy to get confused by emissions categories with cloud providers talking about their own scope 1, 2, and 3 emissions. However, to simplify things, from a cloud perspective (or any outsourced service), 100% of the provider's scope 1, 2, and 3 emissions add up to create your scope 3 total for that provider. Think of it like a nesting doll, as seen in Figure 19-1.

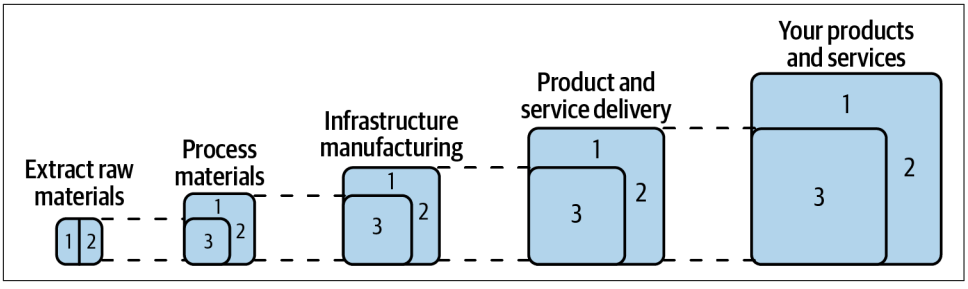


Figure 19-1. Visualization of how different scope emissions stack together and feed into scope 3 emissions as you move up the value chain

Are Cloud Providers Green?

There are two ways to look at this. On the one hand, cloud providers by their very nature tend to operate as efficiently as they can, simply because for them efficiency equals maximum profits, and there is nothing wrong with this. The average cloud provider data center is superior in every way to almost any on-premises equivalent, using the latest and greatest technologies to control usage of precious resources such as power and water.

Most cloud providers use renewable energy where practical, but this isn't enough to make a significant impact on their overall emissions. Cloud providers using renewable energy does not in itself mean they are green or net-zero.

On the other hand, many cloud providers focus their attention primarily on scope 1 and 2. The problem with scope 1 and 2 related emissions is that they are only a small proportion of the total emissions. Factoring in the much larger scope 3 emissions makes the picture look entirely different, with scope 3 often being far larger than scope 1 and 2. Microsoft's 2021 whitepaper "A New Approach for Scope 3 Emissions Transparency" reports that of their 16 million metric tons of emissions in 2020, 12 million of them were scope 3.

Until cloud service providers start providing clearer and more comprehensive reporting on all three scopes of carbon emissions, it is difficult to truly answer the question of how green the cloud is.

The major cloud providers take different approaches when it comes to reporting on their emissions, and they are at varying levels of maturity. So far, the development of cloud sustainability reporting appears to be taking the same path that we saw cloud billing data take in the brief history of cloud billing data from Chapter 5. It started as very summarized, infrequently updated reports and is moving toward more granular, more frequently updated, and easier-to-access datasets. Let's hope it doesn't take a similar number of years to get to the datasets needed for real sustainability measurements and optimizations to happen.

Unfortunately, the current reporting coming out of the major cloud providers on their carbon emissions is murky, with a lack of consistent methodologies and completeness. For example, at the time of this edition, reporting from AWS and Google in this area covers only their scope 1 and 2 emissions, ignoring their scope 3. That leaves Azure as the only provider reporting carbon emissions across all three scopes, with all making extensive use of averages and differing ages of carbon data.

If you read the latest media releases from the major cloud service providers, you would be forgiven for assuming that measuring the carbon footprint of your cloud is a solved problem, with the top three cloud providers now supplying some form of carbon report. While this is a step in the right direction, there are still a number of issues that need to be addressed for the sustainability of your cloud footprint to be understood.

Access

While the top cloud service providers have released sustainability tooling and/or carbon usage reports, many of the other cloud providers either have no capability to offer data on sustainability or require their consumers of cloud to pay for contracted engagements to perform a point in time audit. The frequency of updates to sustainability data ranges from manual (upon request) to monthly. The data itself is delivered in either PDF reports, custom dashboards accessible via the cloud platform web interface, or, in limited cases, API data for use in your own analytics reports.

The lack of standardization of available data and access methods has limited the sustainability capabilities offered by third-party vendor platforms. One option in the open source space is an application called Cloud Carbon Footprint. Cloud Carbon Footprint uses your cloud service provider billing data and sustainability data made available by your cloud service provider. From the data collected, a carbon footprint estimate is generated for your cloud usage. It is an innovative approach to the problem of reporting, and provides interesting visualizations, but it is reliant on the validity and maturity of data and calculation methods available, including the data provided by the cloud providers.

Completeness

You will face difficulties when trying to understand the completeness of sustainability data provided by cloud service providers. Most of the difficulty in understanding the data is driven by the methods used by the cloud service providers to calculate the carbon footprint. Missing scope 3 was mentioned previously, but not all services offered by the cloud service providers are included in their reports, and large amounts of averaging are applied to the calculations. The challenge with averaging is that it can provide a misleading carbon footprint for differing regions, data centers, or individual products. For example, a low carbon instance could appear to have the

same footprint as a relatively inefficient, aged alternate instance. Some cloud service providers currently suggest using cloud spend as a direct proxy to the carbon impact, which is inaccurate—more expensive does not always mean more emissions.

To build trust in the data provided, the cloud service providers need to include all cloud services, be more forthcoming on their methods of calculations and covered gases, and reduce the use of averages.

Granularity

Most of the data being provided by cloud providers is aggregated to a country or geographic region, making the decision—based on the sustainability—between deploying cloud services to one cloud region versus another impossible. None of the cloud providers supply truly accurate carbon data at a cloud resource level today, instead summarizing the data to total carbon emissions per cloud provider service. Without resource-level data, you can't determine which resource types are more efficient than others. For example, it is not possible to identify within a region that a specific location is more or less sustainable or carbon efficient, nor can you compare instances or services with any level of accuracy between regions or locations.

Currently, the cloud service providers do not provide emissions calculators like they do with cloud costs calculators. Without a method of proactively calculating the expected emissions of a workload, the only way for your organization to know the impact is to deploy the service and then wait for updated carbon reporting.

Don't let perfection be the enemy of progress. Though there is plenty of room for improvement on sustainability data that is made available today from the cloud service providers, waiting for all the right data only prevents your organization from developing the culture and skills required to optimize and make decisions based on the sustainability of your cloud deployments.

Partnering with Engineers on Sustainability

An emerging trend is the related—but different—practice of GreenOps. The goal of GreenOps is to build a practice engaging engineers with the sustainability of their cloud. *GreenOps* is used to describe the practice of organizations driving up the sustainability of their cloud deployments and other areas of IT such as data centers, infrastructure platforms, and software development.

Recent extensions made by AWS to their Well-Architected Framework introduced a sustainability pillar, which outlined a shared responsibility model on sustainability. Inspired by this, Figure 19-2 was created to help explain the split between the two key areas: the sustainability *of the cloud* and sustainability *in the cloud*, with the former

being the responsibility of the provider (i.e., Microsoft, Google, or Amazon) and the latter falling firmly in your lap as the consumer.

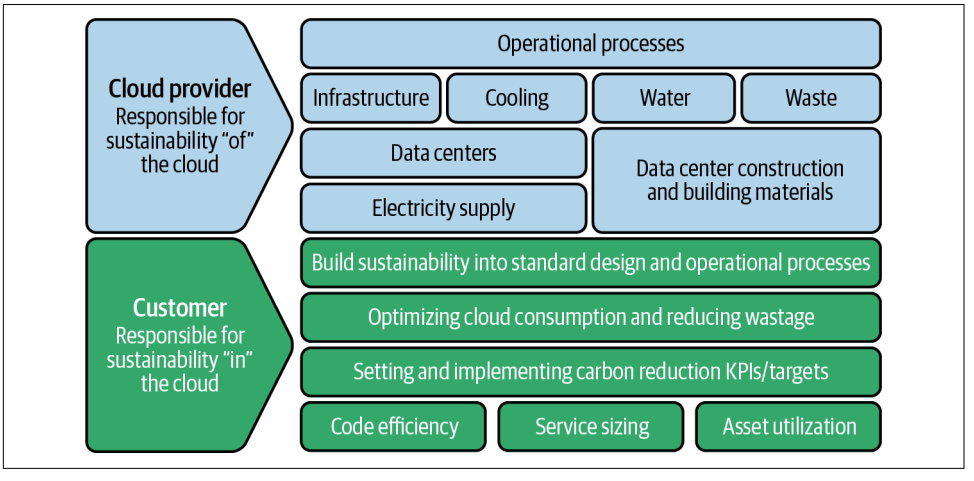


Figure 19-2. The cloud sustainability shared responsibility model (contributed by Mark Butcher)

Sustainability *in the cloud* is a critical component of a sustainable cloud strategy and in general is directly aligned with the principles of FinOps. Put most simply, it reflects the need to create sustainability as a *nonfunctional requirement* (NFR) that becomes embedded throughout your IT organization, making sustainability a key consideration alongside other areas, such as accessibility, security, performance, cost, and availability.

It may not always be possible to make the most sustainable decision when deploying resources in the cloud, but it is important that sustainability becomes a key criteria that is at least considered and that decisions are documented and recorded. As we discussed in Chapter 1, the Prius Effect helps you make good decisions to optimize when you have the information on your use in front of you, and it allows you to explicitly spend, or use more when it is justified.

FinOps and GreenOps Better Together?

In many ways, sustainability requires adding one more metric to what you are already doing within FinOps. The aim is to add the environmental impact to the decisions being made by your organization in the same way you have introduced costs. This is not suggesting that the FinOps practitioner is the new GreenOps practitioner, but rather that much of the work getting cost data into the decision-making process is the same that will need to be repeated for GreenOps with carbon data.

A well-orchestrated FinOps strategy, designed for cost optimization, will expand GreenOps as well. The reverse is also true; GreenOps strategy can positively influence workload efficiencies, resulting in cost savings.

—Bindu Sharma, Senior Manager of FinOps at Guidewire, from her FinOps X 2022 talk

Imagine for a moment the decision process that happens when your organization considers which region to build a disaster recovery environment. From the engineers' perspective, they will take into account latency, recovery point objectives, and recovery time objectives. The engineering metrics are considering the impact to the customer in the event of a disaster. FinOps introduced the cost of operating in different regions so the business can make a decision that considers the business financials, and now GreenOps brings the environmental impact and carbon cost of the regions being considered to help advise how the decision impacts the organization's sustainability goals.

In all the ways FinOps has been working to get its cost data in the path of engineering decisions, GreenOps will need to do the same thing with sustainability data. Instead of the team implementing GreenOps having to go down the path of figuring out who to work with and how to implement the needed changes to these existing systems and processes, you should consider partnering with the FinOps team to share knowledge and possibly aid in the deployment of sustainability data.

Having FinOps help GreenOps get established in your organization is mutually beneficial. Many staff care deeply about the environment, and sustainability data can be the needed motivator to tip the action scale—covered in Chapter 3—in favor of driving action on FinOps optimizations.

Build a benchmark of your current footprint. Document any assumptions you're making and articulate any gaps. Consider opening a sustainability steering group helping to drive the culture of accountability around how each team's actions impact the organization's carbon emissions. Once you have the baseline, then you can start assessing where there are potential carbon savings. Assessing any hot spots of carbon waste and building a culture of remediation, with a roadmap to begin making changes.

—Mark Butcher, Founder, Posetiv Cloud at FinOps X 2022

Normally, sustainability can cost the company additional money, but with an established FinOps practice it's possible to reduce your cloud carbon footprint while also reducing your cloud spend. There is obvious alignment between the world of FinOps and GreenOps with the elimination of cloud waste by reducing usage of unnecessary or unoptimized resources. With all the advancements in cloud native and third-party SaaS FinOps platforms, you can spot an underutilized EC2 a mile away. What is harder to see, for example, is that the way that EC2 was deployed isn't sustainable. FinOps can work alongside GreenOps to optimize the processes and practices of using the cloud versus just optimizing the resources in the cloud. It is important when deploying workloads that you consider sustainability in your decision-making

process when selecting a region or resource type as well as other key criteria such as cost, performance, security, etc. This may require reviewing information sources outside of your cloud provider, such as carbon intensity metrics from sources such as Electricity Maps, where you will find near-real-time carbon intensity data points.

At the very least, the FinOps team can help the GreenOps team to get established and to become better connected to the various organizational stakeholders and can share taxonomy, tagging, and reporting structures that are in place. Just as the FinOps team sought help to integrate reporting into any preexisting ITAM, Software Asset Management (SAM), Security, or TBM groups, it can now repay those favors to advance GreenOps. To get started with collaborating with your GreenOps team, see Chapter 25, which provides a set of guiding principles of how to work with related technology frameworks.

GreenOps Remediations

From a FinOps perspective, there are some well-known areas to start with when looking at achieving cost savings, and the same is true for GreenOps. The following list is not exhaustive but is a good starting point for any FinOps professional wanting to ensure that their services are both cost-effective and sustainable:

- Oversized compute instances
- Orphaned instances left running
- Services running on aged, inefficient instances
- Incorrectly sized microservices
- Compute instances not terminated, or development environments left running
- Excessive retention of storage snapshots
- Data stored on the wrong tier of storage
- Services running in the wrong region/availability zone

Now, there's obviously lots more, and this list is barely the tip of the iceberg, but a critical first step is to assess where you are today and build some sensible time-bound KPIs for optimizing your services based upon actual metrics. There's also a lot you can do to work with your engineering teams to optimize code. Doing that can offer some stellar benefits, but it obviously needs a lot of thought, planning, and engineering time. Automation—discussed in Chapter 22—can help with managing these areas of waste not only for the benefit to FinOps, but also for GreenOps.

Avoid FinOps Working Against GreenOps

Not all FinOps capabilities are aligned with the goals of GreenOps, and actually some of them actively work against the goals of GreenOps. Take, for instance, *rate optimization*, committing to cloud resources to save money, which doesn't help decrease carbon emissions. Cost and sustainability may go in different directions. Commitments to reduce rates on underutilized resources will improve cloud cost efficiency but could also limit your ability to reduce cloud usage and in turn to reduce carbon impact. You may be saving money, but rate optimizations on unused resources may encourage you to keep them around instead of turning them off. This could prevent you from reducing carbon emissions on cloud resources that you may not truly need.

When comparing regions to operate cloud services, the cost does not always match the sustainability. Table 19-1 shows a comparison of a single *e2-highcpu-32* compute resource running across different Google Cloud regions. The data has been sourced from the Google Cloud pricing page and Google Cloud Carbon Energy page. Comparing regions in Table 19-1 based purely on price has Iowa, South Carolina, Columbus, and Oregon as all equal at $578 per month. When you consider sustainability in addition to cost, the choice between regions becomes more favorable for Iowa and Oregon, and South Carolina turns out to be the lowest user of green energy. This table also highlights the issue with using averages across a country: with the average amount of green energy being used at 54%, five of the nine regions actually use less than 54% green energy.

Table 19-1. Comparison of the price and green energy used by a single e2-highcpu-32 instance within Google Cloud

Region	Location	% green energy	Price per month
us-central1	Iowa	97%	$578
us-east1	South Carolina	25%	$578
us-east4	Northern Virginia	64%	$651
us-east5	Columbus	64%	$578
us-south1	Dallas	40%	$682
us-west1	Oregon	88%	$578
us-west2	Los Angeles	53%	$694
us-west3	Salt Lake City	31%	$694
us-west4	Las Vegas	21%	$651
Average		**54%**	**$631**

The comparison becomes even more obvious when you view this from a global standpoint. At the time of writing, the carbon intensity of electricity in Norway is 31g of CO_2e per kWh, compared to 620g in Northern Virginia, USA—a 2,000% difference.

A decision on which region to operate services based purely on the financial impacts could negatively impact your sustainability goals in GreenOps. When making sustainability trade-offs, aim for a shared agreement on the balance between cost and sustainability. Aligning the groups will have them working together with engineering teams and avoiding conflict.

Conclusion

This chapter briefly introduced how sustainability practices like GreenOps intersect with FinOps. As the global focus on sustainability grows, we believe that alignment between FinOps and the teams focused on cloud sustainability will become critical, and the alignment between the two disciplines will enable efficiencies that consider both dollars spent and emissions output.

To summarize:

- Sustainability has become very important to organizations, driven by multiple constituencies including government, customers, and staff.
- The cloud can be greener than traditional on-premises platforms, but without all the required data it's impossible to determine how sustainable it truly is.
- Existing FinOps practices are positioned well within organizations to help with GreenOps success by leveraging similar remediation steps, reporting patterns, and cross-functional alignment.
- Granularity of sustainability data would enable FinOps to help teams make more sustainable choices with their cloud deployments.
- Accept the limitations of the currently available data and get moving; avoid waiting for perfection before you start making a positive impact.
- Sustainability is an opportunity for FinOps practitioners to engage more widely across their organizations and can act as an additional lever for your FinOps remediations.
- Consider sustainability as a metric along with cost in your FinOps conversations.
- Use sustainability in your conversations with other teams, such as DevOps/engineering, to help drive a culture of efficiency and optimization.

This concludes the optimize phase of the book. At this point, you should know what optimizations are available to the organization, and you should have set goals for what you would like to achieve. Part IV will continue on to the operate phase of the FinOps lifecycle, where you'll build out processes that enable you to achieve your goals.

PART IV
Operate Phase

In this final part of the book, we move into the operate phase of the lifecycle, where action is taken by implementing processes and automation. We also dive into building partnerships that accelerate FinOps across the organization. Finally, we arrive at the ultimate state of FinOps: data-driven decision making and the secret ingredient to FinOps success.

CHAPTER 20

Operate: Aligning Teams to Business Goals

Without a process for action, goals are never achieved. As you move into the operate phase of FinOps, you work to support and strengthen the elements identified in the previous two phases.

In this chapter, we look more broadly at processes and how they not only enable automation but also help organizations achieve their FinOps goals.

Achieving Goals

Whereas the optimize phase sets the goals, the operate phase takes action. New goals aren't set here—this is where you make decisions and put plans into place to address the goals that have already been set, based on the business case and context.

Let's take the concept of turning off resources out of hours—when not needed—and apply the correct order of operations. During the *inform phase* you identify the resources that could benefit from being turned off. In the *optimize phase* you measure the potential savings and set a target for how much you'd like to save based on these metrics. In the *operate phase* you define the process of turning resources on and off, and you can enable automation to increase the likelihood of the resources having the new process applied. This is the phase in which you introduce scalability and replication by defining processes that can be implemented across the whole organization.

The operate phase doesn't always lead to action. If you think about the optimize phase as where you identify the opportunity for optimization and build the mini–business case for action, the operate phase is where you reach the decision to take, or perhaps *not* take, an action.

> *Most information is irrelevant. Knowing what to ignore saves you time, reduces stress, and improves your decision making.*
>
> —Shane Parrish, Farnam Street

With each iteration of the lifecycle, another goal comes into focus, with new processes and automation being built upon to reach that goal. While building processes you might uncover more goals you would like to achieve. When that happens, return to the inform and optimize phases to correctly measure and set targets for new processes. Keep each iteration of the lifecycle small and quick to correctly measure each change you make.

Staffing and Augmenting Your FinOps Team

Over time, most organizations hire experienced FinOps practitioners to staff their internal teams, as we discussed in Chapter 3. However, the number of new jobs in the space has exploded in recent years, outstripping the available talent pool of experienced individuals on the market. IDC in 2022 estimated that 60% of enterprises already have or are building FinOps teams. There can be fierce competition for talent.

Many turn initially to FinOps consultants to augment and kick-start their teams as they look to upskill internally with FinOps training and attract talent externally. All the major global consulting firms and big accounting firms now boast FinOps practices, including Accenture, Deloitte, EY, KPMG, PWC, etc., and systems integrators including HCL, SADA, SoftwareONE, Virtasant, etc.

There is, however, a big variety in the maturity and size of these practices in each, so be sure to confirm whether they are FinOps Certified Service Providers, and verify the number of FinOps Certified Professionals they staff.

Processes

On their first time through the lifecycle loop, most organizations focus on understanding their current costs. This includes building processes around what accounts they have, identifying how the bill is organized, and implementing some form of spend analytics tooling, either from the cloud service providers directly or from third-party FinOps platforms, to achieve higher-level capabilities like complete chargeback or container reporting.

After this, the most important goal for your organization should drive the processes you build. A cost-sensitive organization will focus on cost efficiency as the most important goal, while an organization focused on rapid innovation will choose to focus on enabling speed.

Many processes work together to formulate a successful FinOps practice. Automation follows process.

Before you can build automation, you need to build out processes for that automation to follow.

It's important to define who is responsible for (or "owns") the process, what parts of the organization must follow the process, and to what goal or goals the process contributes. A clearly defined operating model outlines expectations for staff.

When you adopt third-party automation tools—covered in Chapter 21—they often dictate how the process must work. When using these tools, you may need to adjust your defined processes to suit.

Start by figuring out how items get added to the new process (*onboarding*), what is expected from teams (*responsibility*), how teams are informed about when the process should be followed and the outcomes of following the process (*visibility/alerting*), and finally begin following the process (*action*).

Onboarding

Circumstances change, especially when you're using the cloud. Your organization might add to its existing cloud deployments or go through a company merger. All processes should clearly define what changes with cloud growth; otherwise, you end up with a process that works for the present but not for the near future. Don't just think about the now, think about what the future may bring, and start thinking ahead about which parts of the process will need to evolve so you're setting yourself up for future changes and success. Getting buy-in from the teams that will be actioning the process should happen well before action needs to be taken, and setting expectations with them that there will be changes as the practice matures. Convey to teams the goal and outcomes that the new process is aiming to achieve.

Stories from the Cloud—Mike

At Atlassian, there are specific processes for onboarding new cloud accounts. These processes define how an account is added to the existing billing structures and how FinOps tooling is added to the new account. Governance tooling is added to the account, enabling idle resource management and rightsizing recommendations. Any existing RIs in the account are merged into the centralized management by the FinOps team. Having this governance builds a standard approach to account management, removing the need for individual teams to manage their RIs, and allows for more savings over time. The items being governed will evolve as the FinOps

practice matures, such as when we moved from using Trusted Advisor's compute optimizations to using Compute Optimizer's recommendations. This allowed us to cover more cloud services beyond the basic compute. The central team responsible for governance is constantly working on providing better tooling to work with the next order of magnitude of scale.

Responsibility

Each process should be allocated to an owner. Effective processes clearly define who does what, and when, and how they do it. Do teams follow the process daily, weekly, monthly, or at some other frequency? Remember some processes will be followed by the FinOps practitioner (like buying commitments), while others are distributed (like rightsizing and usage optimization).

Responsibilities assigned to teams will grow as processes mature. Regularly communicating any increases in expectations will reduce the likelihood of teams claiming they were unaware of what they should have been doing and when.

Through each iteration of the FinOps lifecycle, you build on these processes. At the beginning of the cost visibility journey, for example, the only expectation of a team when they receive a cost report might be to read it. Later, teams might be asked to investigate and explain any costs over a certain threshold. Even later, teams might be required to revise budgets and forecasts when projections show goals won't be met.

Once again, you're gradually maturing FinOps. It's impossible to have teams investigate costs during the first time through the FinOps lifecycle without first putting in place a well-developed cost allocation strategy and correctly forecasted budgets. Instead, you minimize time through the loop, because trying to build all the processes at once makes it impossible to track the impacts of each process.

In photography studio and theater lighting design, there's a maxim that reminds the lighting technician to *only move one light at a time*—that way, they can see the impact of each change before making additional adjustments. The same holds true here: if you change processes related to your rightsizing or tagging and to your reserved instance or savings plan management at the same time, you may not be able to easily identify which change drove movement in your bill.

Visibility

Both before and after an action is taken, you must build visibility. In the operate phase, processes that enable visibility ensure that everyone is aware of the changes required and those being made. The most common operational processes

for visibility send automatically generated reports or alerts to teams to make them aware of events as they happen.

Reports should be clear and easy to understand using the FinOps common language we defined in Chapter 4. Training and documentation should be made available to ensure all staff are familiar with this lexicon. Alerting and anomaly detection should occur on predictions based on current metrics, allowing staff to respond before things are off track. Instead of waiting for a target line to be breached, forecast the near term to identify when action is required early. A good example of this is AWS Budgets, which allows alert configuration based on predicted end-of-month spend.

Over time, processes and communication channels will grow in importance and sophistication. The FinOps team can bring development teams, finance teams, and business teams to the table to learn what information can help them plan and make decisions more effectively, as well as to receive information about their plans. Over time, the amount and variety of information provided will increase, and it will be critical for FinOps practitioners to continuously improve how they communicate.

Many FinOps teams start by holding monthly or quarterly meetings with the technology teams building and running applications. The first meetings can be awkward, uncomfortable, and somewhat guarded, with neither side quite understanding what the other wants. Over time, as these discussions continue, the FinOps team works with the IT teams to advocate for them, to evangelize the good steps they are taking to understand and control costs, and to provide them with a better understanding of business goals and finance requirements. Records of these meetings show deeper conversations about complex cloud decisions the team is considering, and participants begin to come to the meeting prepared to talk about ideas to save costs, rather than being dragged through their own cost data.

Combining visibility with clear responsibility and role definition breeds trust and leads to partnership in action. Working with the various teams of your organization is one of the processes the FinOps team needs to build. Be careful not to overlook the processes related to operating a FinOps team successfully when building processes such as rightsizing and commitment-based discount program planning, which are more focused on the cloud providers.

Action

The action stage is where something actually happens—processes involving activities that enable your organization to reach its FinOps goals, from generating new reports to turning actual cloud resources on and off. Action processes are often good candidates for automation. When we discussed rightsizing in Chapter 15, we highlighted that recommendations are often generated by the centralized FinOps team and then distributed to engineering teams so they can investigate and adjust their resources where appropriate. Without clear processes for how rightsizing recommendations are

generated or communication about what is expected from engineers, rightsizing often does not succeed.

Central FinOps groups can be great sources of information on how to perform the various FinOps capabilities at increasing levels of sophistication. Internal knowledge sharing, process repositories, and tools can be a huge benefit, particularly to large or complex organizations. Scripts, process documents, and how-to materials can be collected for use by all, along with records and logs of actions taken or decisions made. Remember, different groups within any organization will be at different maturity levels and different stages of the FinOps lifecycle. They should have a clear place to go to find processes that work.

A direct action might not always be the decision you reach. Even if you have an excellent business case for rightsizing, it might not be feasible if critical staff members need to focus on another strategic initiative. You might postpone a commitment purchase by two weeks to accommodate a pending decision about cloud architecture, which could radically change the demand for the services being considered. Not every optimization opportunity will lead to an action or a concrete saving.

It's just as important, though, to document these decisions and the rationale for making them, to provide a record of action and guide future decisions.

How Do Responsibilities Help Culture?

Processes that define which teams are responsible for taking actions also help build the culture around how different teams work together. Teams should understand how their contributions help the organization reach goals. When teams don't see how their efforts contribute, they often don't follow processes.

Carrot Versus Stick Approach

Reporting that helps highlight the positive impacts of following processes encourages teams to put in the effort to realize these benefits. It's also important to track and report where teams *haven't* been following processes and measuring those impacts. Building out the best performers and worst offenders list can help the organization to reach its goals. How these reports are distributed inside an organization will be determined by the culture of the company. Some publicly reward success and privately deal with teams that don't apply enough effort, while others prefer a public "worst offenders" board.

While traditional IT organizations typically work with positive metrics rather than negative ones, in the world of public cloud and FinOps there's one great distinction: teams have the power to stop waste and negative behaviors in a way that they never did when running in the data center. Punishing a team for wasting storage capacity on a storage area network (SAN) whose cost was long ago depreciated serves little

purpose, but highlighting unused resources that continue to cost the company money hour after hour, despite the fact that they could be terminated with a few clicks, makes sense.

In usage optimization, a common issue is getting teams to implement changes. The ability to show the overall savings your teams can achieve and what savings other teams have generated will help convince teams to engage with the process.

The preference here is to use the *carrot* as a way to build collaboration with engineers and avoid creating an *us versus them* environment. Unfortunately, the aforementioned carrot is not always enough, and you will need to work with teams that are not following the guidance from FinOps.

Remember that some teams are expected to spend significantly more than others. A "worst offenders" list based on total spend won't be productive. While it might be topical that team A spends more than team B, it's more important to know if team B spends more than budgeted while team A is on target. Focusing the conversation on the things that are critical, and not wasting time on those that aren't, plays into the language of FinOps.

Handling Inaction

It's important for the executive team to propagate the message that meeting targets is important to the success of the whole organization. When teams understand this, they pay more attention to reporting on their overall performance. When teams want to sacrifice spend for speed, management needs to sign off. This exception should then be factored into budgets.

It's not often that you come across a staff member who doesn't care that the organization is overspending. So when we find that our teams aren't following the advice of the FinOps team, it's most often because they have other priorities. Understanding what these priorities are can avoid conflicts of opinion.

Teams might focus on the speed of execution more than the cost. Getting a project done earlier might give the business an overall market advantage. If a team is having problems with what they've deployed, stopping them from addressing that in order to save money might not be in the business's interest.

If team members are reluctant to allocate time to meeting targets due to other project timelines, then either the management responsible for the roadmap planning needs to allocate time specifically to optimization, or the budget lead needs to capture the increase in spend due to time not being allocated to optimization.

In cases where there is good reason to be over target, details might be missed in the target-setting phase. Spending some more time target setting could prevent a team from appearing in the "worst offenders" list.

What do you do when a team isn't prioritizing addressing their targets and isn't willing to make the needed changes? That's when a FinOps practitioner advises the business unit's budget lead of which targets are being missed and what opportunities are available to help the team get back on track.

Putting Operate into Action

Let's look at an example of usage optimization by removing idle resources. The FinOps team defines a process of detecting idle resources and estimating the potential savings. During the optimize phase of the FinOps lifecycle, the FinOps practitioner measures the amount of idle resources.

Assume you have $150,000 in potential cost avoidance, of which a goal of $50,000 is set. You define a process that assigns responsibilities and clearly outlines what is expected from engineering teams, how quickly they are expected to respond to the recommendations, and what teams should do to exclude their resources from the process. You generate reports for these idle resources and publish them to the teams responsible for the resources.

As teams remove idle resources, the amount of costs that could be avoided decreases from the initial $150,000. Engineering teams will see the number of recommendations assigned to them coming down as well. Both results help show the impact of the teams' efforts.

Where teams aren't following the process, their recommendations and potential savings will not change, or worse, they will increase. Management should reiterate the importance of following the processes to remove idle resources, putting pressure on teams that are not applying the process. The FinOps team should collaborate with teams not meeting their goals to determine how they can be assisted. This is covered in more detail in Chapter 24.

As the goal of $50,000 in costs avoided is reached, new—and hopefully more ambitious—goals can be defined.

Conclusion

To achieve goals, every organization needs processes that clearly define who is required to do what and by when.

To summarize:

- Set goals in the optimize phase, and build processes in the operate phase to achieve them.
- The FinOps maturity approach applies to processes. Don't try to achieve everything all at once.
- Developing processes in small increments allows you to measure progress toward your goals.
- Management buy-in can help with teams not completing their required tasks.
- Collaborate with FinOps practitioners outside your organization to adopt processes already proven to work at other organizations; consider joining the FinOps Foundation, using FinOps contractors, or hiring experienced practitioners.

There are many processes that lend themselves to automation: report generation and alerting are two primary candidates. In the next chapter we discuss how automation in FinOps should follow only a clearly defined process.

CHAPTER 21

Automating Cost Management

When you have a repetitive task that needs to be performed often, automation removes the effort and maintains consistency. Within FinOps there are plenty of opportunities to automate. In this chapter, we'll take a look at some of the commonly automated FinOps tasks and the benefits and challenges of automation.

FinOps automation as a term can mean many different things, ranging from simple bill processing to direct infrastructure management, including processing of billing data, anomaly alerts, report delivery and creation, reserved instance or savings plans management, tagging/account/subscription/project creation and hygiene, usage optimization and rightsizing, stopping and starting of resources on a schedule, predictive autoscaling based on demand, choosing instance sizes on deployment, migrating workloads between spot and on-demand, or delivery of cloud spend data to external systems, etc.

Your definition of automation will depend on your company's cloud and FinOps maturity, its staff's technical abilities, and your general comfort level with moving repetitive processes from humans to computers.

Whether to Automate

To automate or not to automate? That is the question. You can determine the answer by looking at two aspects of your organization. First, you must outline the outcome you want to achieve with automation. Second, you need to compare automation to a manually tracked process within the organization and determine whether automation is a better method to achieve the desired outcome you've identified.

Should you automate? Ultimately, the answer is "it depends," as is often the case. To answer the question, we must look to the benefits achievable from automation.

What Is the Outcome You Want to Achieve?

It's important to identify an actual business outcome, not just the way that automation will achieve that outcome. For example, if you take another look at idle resource governance, you could be forgiven if you think the goal is to *remove idle resources* from your cloud environment. But in reality, the outcome is that you want to *remove the costs* of these resources because they're costing you money. This is important, because you use such goals to measure the performance of your automation.

If you measure the raw number of resources turned off—since they're identified as idle—you won't see the real impact this has on your cloud bill versus when you measure the costs avoided by shutting these resources down. If you develop the discipline to automate wasteful usage, you help protect yourself against sticker shock down the road. You also get your organization into the healthy habit of thinking through where waste in new cloud products may emerge.

As another example, tag governance aims to drive adherence to your tagging standard. It does this by removing resources or alerting you when tags don't match your standard. You will measure the impact of your automation by tracking the number of resources that do not meet your tagging standard over time.

With scheduled resources, you also want to lower your costs by reducing the usage of resources during your out-of-office hours.

Automated Versus Manual Tasks

On the surface, this might appear to be a silly exercise. After all, doesn't everyone want to automate everything?

There's a benefit to stepping back from the goal you are trying to achieve and validating that automation is the correct solution, though. If your goal is to save money, then you should take a moment to compare the potential savings your proposed automation could generate against the cost of buying or building automation, with a keen eye on the internal security perception considerations of using an outside tool as well as the management considerations of maintaining an in-house one.

If you're going to use a third-party platform to automate the task, you should examine the ongoing costs. If the planned automation won't save your organization money in a reasonable timeframe, or if the cost of maintaining the automation will exceed the savings potential, then there's no discernable business benefit to deploying that automation.

Keep in mind that sometimes our goal for automation *isn't* to save money. This is often true with tag governance, where you want to drive the adoption of your tagging

standard, and ultimately increase the accuracy of your showback and, really, to drive team accountability for cloud spend.

In all cases, seek to validate that automation will help you deliver on your goals. For example, if alerting on invalid tagged resources isn't combined with a corporate policy to react to these alerts, then you may well decide that automation is just adding noise to everyone's day without delivering on the goal of increasing compliance.

For small-scale or slow-changing cloud footprints, deploying and managing a tool to automate alerts might require more effort than just having someone follow a manual process. But as your cloud footprint grows and/or becomes more dynamic, automation becomes more important. Remember, as your team has more and more tasks to deliver, manual processes tend to fall down the list or are forgotten completely. In these cases, automation ensures that the tasks are completed in the same manner and on schedule.

Sometimes the effort of automating—which may exceed that of doing a task manually—can be worth it. Humans make mistakes, and automation can be used to correct them. Humans might mistype a tag's case-sensitive key/value pair. Automation doesn't make that mistake, and it has the added benefit of reducing a team's efforts. A great example of using FinOps automation to reduce errors and effort is when you automate the process of adding supplementary tag data to resources. Teams may tag their resources with a simple identifier and then have automation add extra, clarifying tags based on the correlation between that identifier and an outside data source like a CMDB. It's common for teams to use a project identifier that maps to cost centers or to environment types, or to have the project identifier tag itself be automatically identified from the outside CMDB based solely on the identifier. Automation would find that project identifier, add the extra tags to the resources, deliver the relevant cloud spending data to that team's systems for display or future automation, and generate cloud spend reports for that team such as anomaly alerts. This reduces the number of tags teams need to apply, avoids human error, and makes sure that the extra tag data is applied in a consistent manner, then delivers that data into the path of the engineers who need to see it in the places they are already working.

The ability to have a conversation with the business that's not just about *how* to use automation to deliver a goal like cost savings, but also about *why* automation is the correct way to solve an issue, allows you to more thoroughly reason with the costs of implementing and maintaining the automation.

It's important to reassess automation's benefits and costs repeatedly as you cycle iteratively through the FinOps lifecycle: what initially didn't make sense to automate may prove worthwhile once you hit a certain scale, or conversely something you did

automate initially may be more complex to manage or troubleshoot than expected, causing a temporary move back to a manual process. As your cloud footprint grows, as free and open source tooling becomes more widely available, or as you implement tooling for other reasons that offer some automation as part of a package, you may find the cost-benefit analysis favors adding automation.

Automation Tools

When it comes to any business tooling, the inevitable debate begins: buy versus build. For FinOps, we recommend starting with a third-party tool; initially this means native tools from the cloud providers, and later SaaS providers once your spend size and complexity warrant them. Successful homegrown tooling generally needs to come from a deep understanding of what works and doesn't for your organization gained from using a range of available commercial solutions.

When you begin your journey into FinOps automation, you're likely to repeat the same process of learning that many have followed. By using well-established third-party tooling, you can avoid some common pitfalls, accelerate delivery time for automation, and then develop an understanding of what works well for the organization and what doesn't. This is all part of the same *Build versus Buy versus Native* conversation we introduced in Chapter 8.

While build versus buy is dependent on the company, we also believe that you shouldn't try to build your own FinOps solution without having hit the limits of the available ones, a journey that enables you to understand the complexities inherent in FinOps processes and their battle-hardened tools (whether native, third party, or open source). In our experience, it's a fool's errand to build new FinOps tooling as a new practitioner, regardless of how awesome your team's engineering chops are.

Costs

When thinking about costs versus benefits, the old axiom of "you don't know what you don't know" definitely applies. There are plenty of knowledgeable companies and contractors that can share their insights on what practices and processes need to be implemented around FinOps automation. Indeed, they can guide your whole organization to realize the most benefits of the tooling they offer.

It's also important to be realistic when you consider what you can build compared to what you can buy. It's easy to fall into the trap of comparing the minimum you need to build to get the task done. However, this leaves you with inadequate information about how the tool you build will run, and what impact it will have when things aren't working as expected. A third-party tool will very likely come ready to show its

benefits and value, and it will have a team of people whose main focus is maintaining and improving the tool—so your teams don't have to.

Other Considerations

Cost isn't, and shouldn't be, the whole story when it comes to buy versus build. There are security implications. Also, you might find that the features a tool offers aren't compatible with your environment or can't be integrated with something you already use like chat and ticketing applications for alerts and notifications. Some industries even have specific compliance requirements like FedRAMP, HIPAA, SOC 2, or PCI, and the third-party tooling providers may not currently hold the right certifications. Finding a vendor with the right set of certifications can make your automation process easier from a compliance perspective.

Being able to audit automated actions is key, especially when the automation interacts with production environments. This knowledge builds transparency.

Third-party automation tools are usually well documented and provide training materials for staff to learn from, so when you're hiring new staff it's more likely that they are already familiar with the tooling. If you build automation tooling yourself, you need to be willing and able to provide ongoing training for existing and new team members, and you run the risk that those responsible for building and maintaining the tooling may leave the organization. Unless there are more than a few employees familiar with the tooling and its implementation, you encounter "key-person risk."

When you decide to build automation tools instead of buying off-the-shelf offerings, you should do so in an informed way. You must always compare the effort to build all the required features—including ways to measure the effectiveness of your tooling, ongoing maintenance, and updates you will need to implement—with a solution you purchase. And you should be careful to identify the missing features or compliance requirements of any third-party vendor tooling.

Tooling Deployment Options

As with most software, there are a few considerations around how automation tools are deployed that will have an impact on the selection process.

Cloud native
: With automation that runs within a cloud account, scripts can be provided by the cloud service provider or the broader community. These are executed inside your cloud account in Function-as-a-Service platforms (like AWS Lambda).

Self-built
: This is software that your own staff members build and run. Feature development is run by internal teams. If you have many different DevOps teams around

your organization, you might find they each solve the same problem using their own version of automation. But it's best if a centralized team runs an automation tool that services your whole cloud environment. Centralized management of a tool will reduce duplicated effort around the organization, but it also means you need a one-size-fits-all solution.

Third-party self-hosted
: When it comes to prebuilt automation solutions, there are free and open source tools available, as well as paid software licensing options. With this deployment model, teams use third-party software but run it within their own environment, maintaining and managing the service themselves. In the open source space, a commonly recommended tool is the Cloud Custodian application created by the team at Capital One. You'll want to consider the cost of operating and potentially customizing third-party self-hosted solutions.

Third-party SaaS
: This option removes the need to run and manage software altogether. A third-party hosted software solution gives you the easiest setup and potentially the fastest time to automation. A vendor will often provide simple tools that will handle the setup required in your cloud account for the vendor's platform to integrate with your environment. Complex metrics and configurations have already been developed and often come attached to easy-to-use dashboards and setup screens. You will often pay a monthly fee, which is easy to measure against the benefits of the tool itself. This model does come with a security impact, which we'll cover in a moment.

It's not always simply build versus buy; some companies take a hybrid approach of *buy and augment*, where they combine an off-the-shelf tool's API data or recommendations with their own in-house automation so they can avoid building the data ingestion/management capabilities of FinOps but can own the code that actually touches their infrastructure.

Jason Fuller from HERE Technologies shared that while he uses a vendor FinOps platform to drive his reporting and recommendations, he relies on heavily customized internal automation code (based on an in-house fork of the open source tool Cloud Custodian) to actually effect changes in his environment. This hybrid implementation of build versus buy is a three-pronged (vendor + open source + custom code) approach that he has honed after 10 years of doing FinOps at four large-scale cloud spenders.

Automation Working Together

Automation tools rarely operate in isolation. It's likely that multiple tools will be operating alongside each other, especially over time as your cloud footprint grows.

Integration

Integration between tools can be very powerful, and it's something you should be looking for and taking advantage of. Avoid having many separate tools that do their own version of the same thing. For example, if you have an existing ticketing system at your organization, you would ideally implement FinOps tooling that will generate tickets for action items in that existing system instead of building in a separate task-tracking system.

If the automation tooling you have selected doesn't have direct integration with your other software tools, it's still possible that the automation tooling supports extensible event processing. Then you can receive notifications about events generated inside the automation tooling and can build an extension that will connect the event to some of your other software packages. It's a great way to extend functionality—for instance, sending notifications to your chat application that particular tasks have been done, generating tickets in your ticketing systems for necessary follow-up tasks, or implementing your own automated remediation tasks from the detections.

Automation Conflict

When you deploy multiple automation tools, you need to make sure that they don't conflict with one another. If one tool is trying to start a server while another is trying to shut it down, you can end up with the automations fighting each other's actions. If you are deploying an automation tool for a particular feature, and that same tooling has another feature that you don't intend to use, consider whether you can remove the service's permissions for that disabled feature to perform changes on your cloud accounts.

We've seen companies deploy different automation tools for different features they offer, only to find that both tools are trying to control the same resources. Or one team will implement their own automation that conflicts with existing automation.

This is a harder problem to solve. To prevent it, *you need to educate the whole organization about what automation is in place*, and have teams work together to ensure they understand the possible impacts of any planned automation within their cloud accounts.

Ideally, automation should notify you when it's fighting with something external. For example, if automation stops a server instance more than a set number of times in a short period of time, it should send an alert and prevent any further changes.

Sometimes the conflict an automation service has isn't with other tooling, but with people. You may have automation that stops a server instance while one of your team members requires access to that server. This can lead to the team member starting the instance, only to see the automation stop it. Once again, you should be sure that your team has been well educated, including on how to exclude a resource from the automation.

Joe Daly, formerly Director of Cloud Optimization at Cardinal Health and Nationwide, tells the story of how he accidentally took down a production environment by manually removing a nonproduction resource tied via automation to the production one. As stated earlier, education about automation being used is critical in the organization. To hear the full story, check out Episode 13 of the FinOpsPod podcast, titled "FinOops—Lessons Learned the Hard Way."

Safety and Security

Security is a primary concern with any FinOps automation tooling. Often, FinOps automation requires very high levels of access to your cloud environments, whether it's just the ability to describe how everything in your account has been configured, or the ability to change (start/stop/delete) the resources themselves. Such access could lead directly to a denial of service attack, data loss, corruption, or a confidentiality breach.

Security is often cited as a major reason third-party tooling isn't used inside organizations. It's understandably difficult to get a security team to sign off on the use of a third-party tool that needs broad access to cloud accounts. However, just because your own teams write the software doesn't necessarily mean it will be perfectly secure. You must ensure that all automation tooling you deploy has the least amount of permissions possible to perform the tasks required, while protecting the software from external threats.

It is also highly likely that a successful third-party vendor has spent a lot of time and effort on their security. Reviewing their documentation and any previous security announcements from them will help you gauge how seriously that vendor takes security as part of their offering.

If you look for inspiration from the FinOps maturity model, you might choose to initially integrate with a third-party SaaS tool in read-only mode. You might then take the notifications from that tool and automate the remediation actions using your own tools. When you have more confidence in the vendor, you can start allowing them to perform more actions on your behalf until you have enabled all the features you require.

How to Start

You shouldn't try to implement too much automation all at once. This will inevitably cause issues and have you struggling to make sure the automation you deploy is working effectively. Here are some tips:

Start simple
: Start with a simple selection of tools and automation goals. Trying to build a very complex set of automations and/or coordinating multiple sets of tools all at once will often lead to unnecessary challenges.

Use it in an inform mode first
: Start initial automation in an inform mode, automating the discovery and alerting of potential issues, but not their remediation, so it lets you know what it would do if it were enabled to make changes.

Build confidence in the automation
: Learn about the actions an area of automation will take once it is enabled. Build confidence by sharing the results with teams to ensure they are comfortable with the introduction of automation.

Do plenty of testing
: Once you've built confidence with automation, start enforcing actions in dev/testing accounts first and then test with smaller groups before broadening to a wider user base.

Don't build it all yourself
: Rely on commercial or open source tools. Not only will this save time, but it also will make sure you get the latest battle-tested solutions.

Measure the performance
: Automation should be measured to ensure it's having the desired effects. As automation is scaled out across the business, it's essential to measure that the performance is not degrading.

Once again, we recommend joining groups like the FinOps Foundation that allow you to connect with experienced practitioners. Learn about what tools they use and the benefits they gain.

We can't stress enough the importance of starting small. Remember Gall's law that cogently states that "all complex systems that work evolved from simpler systems that worked."

What to Automate

Successful FinOps practitioners have many automation tools at their disposal. Some automation has to be very specific to the way their organizations operate. But there are some common automation solutions that we have seen implemented over and over by organizations with successful FinOps practices.

Tag Governance

Once you have a tagging standard defined for your organization, you can use automation to ensure it's being followed. You start by identifying the resources with missing tags or incorrectly applied tags and then having those responsible for the resources address the tagging violations. You then move to stop or block resources to force action by the owners, and possibly work toward a remove/delete policy for these resources. However, deleting resources is a high-impact automation, so you don't see many companies getting to this level. We recommend not moving directly to deleting offending resources without working through the earlier, lower-impact levels of automation.

Scheduled Resource Start/Stop

Resource management and automation enables you to schedule resources to be stopped when not in use (e.g., while you're away from the office) and then have them brought back online when you need to use them again. The goal for this automation is to minimize impact on teams while realizing large amounts of savings for the hours that their resources are shut down. This automation is often deployed into development and test environments, where the absence of the resource isn't noticed outside of business hours. You should ensure implementations allow team members to skip a scheduled activity, just in case they need to keep a server active while working late. Also, canceling a scheduled task should not remove the resource from the automation altogether, but should just skip that current execution.

Usage Reduction

Chapter 15 covered items like resource rightsizing and idle resource detection. Reduction automation removes such waste, or at least sends alerts to responsible staff members to drive better cost optimization. Having automation pull in resource data from services like Trusted Advisor (for AWS), from a third-party cost optimization platform, or from resource metrics directly gives you an easy way to send alerts to the team members responsible for the resources to investigate or, within some environments, to autostop or resize the resources.

Conclusion

Automation provides you with an opportunity to increase the reliability and consistency of processes, checking and alerting on FinOps practices and processes. It's important to understand the true goal that you're trying to achieve with automation and to realize that automation is not free.

To summarize:

- Start small, measure the results and benefits achieved, and iterate slowly on your automation, adhering closely to Gall's law.
- Costs of automation come from the purchase or build of software; the resources consumed by the automation itself; and any efforts to manage, maintain, and monitor the automation.
- By setting out with a clear goal you hope to achieve from your automation efforts, you can measure the costs of automation versus the potential business benefits.
- Consider purchasing a well-regarded solution from a third-party SaaS or software vendor before heading down the path of building your automations from scratch.
- Before implementing an external solution, however, you must consider security and feature-oriented implications for your cloud environment.

In the next chapter, we explore an advanced FinOps process that enables just-in-time optimizations using metrics and alerts.

CHAPTER 22

Metric-Driven Cost Optimization

Metric-driven cost optimization (MDCO) is the driver that measures potential optimizations and then uses goals and target lines to trigger an operate process to action them. When you are buying commitments, you're in the operate phase. With MDCO, you use the metric threshold to determine precisely when you take those actions.

MDCO could also be described as "the lazy person's approach to cloud cost management." The primary rule of MDCO is: don't do anything. That is, don't do anything until you have a metric that measures the impact of your actions. If you make optimizations but don't have metrics to tell you whether they've been positive or negative on your spending, you're not doing MDCO.

By the end of this chapter you'll know how to use metrics to correctly drive your cost optimizations.

Core Principles

There are a few core principles that define the MDCO practice:

Automated measurement
: Computers perform the measuring, not humans.

Targets
: Metrics without targets are just pretty graphs.

Achievable goals
: You need a proper understanding of data to determine realistic outcomes.

Data driven
: Actions aren't driving your data: the data is driving you to take actions.

We'll explain each of these principles throughout this chapter.

Automated Measurement

Loading billing data, generating reports, and generating recommendations for optimizations are all tasks that should be done by automation, or via a FinOps platform. When massive amounts of billing data is delivered by your cloud service provider, all of these activities need to kick off automatically. Having FinOps practitioners trying to run ad hoc data processing and report generation slows down MDCO and will result in slow responses to anomalies.

Repeatable and reliable processing of the data allows you to remain focused on the real task at hand: saving money. You can then spend time refining the processes being followed and working on more deeply understanding the cost optimization programs and how they are reflected in the data/metrics.

Targets

For MDCO, target lines are key. We wrote in Chapter 14 about how target lines give more context to graphs. We've always been firm believers that no graph should be without a target line. Without it, you can determine only basic information about the graph itself, not what the graph means for your organization. The target line creates a threshold from which you can derive a trigger point for alerts and actions.

Humans don't do well understanding the scale of very large numbers, which is why contextual targets are so important. To counteract this, a tip from Hans Rosling's book *Factfulness* (Flatiron Books, 2020) was quoted by Anders Hagman from Spotify at a recent FinOps Foundation monthly summit. In the book, Rosling tells readers that "whenever someone gives you a large number, ask for another number" to measure it against. In MDCO, for your metrics to provide value, you need a target to measure against.

Achievable Goals

There are several metrics to monitor across cost optimization programs. Having each metric correctly tracked enables the operational strategies for your cost optimizations. Every individual optimization will have a different impact on your savings realized, so you shouldn't treat all optimizations as equal. When combining metrics, you should be normalizing the data based on the savings potential of the optimization.

One of the most common incorrectly measured metrics, and one that can help you understand achievable goals in MDCO, is *commitment coverage*.

There are multiple ways to measure cost optimization performance, and each method represents a particular view on efficiency. Depending on the method you use, 100% optimization may not be achievable. For MDCO, measurements that are completely achievable are required to determine the right time to perform actions.

Let's work through commitment coverage to explain this in more detail.

Commitment coverage

Commitment coverage is the metric that shows how much usage is being charged at on-demand rates versus discounted rates. Generally, organizations set a target coverage—commonly, it's 90%. Let's compare the various methods.

In the traditional model, you measure coverage by counting up all the commitment hours that you have and then dividing that number by the total hours you ran in the period (see Figure 22-1).

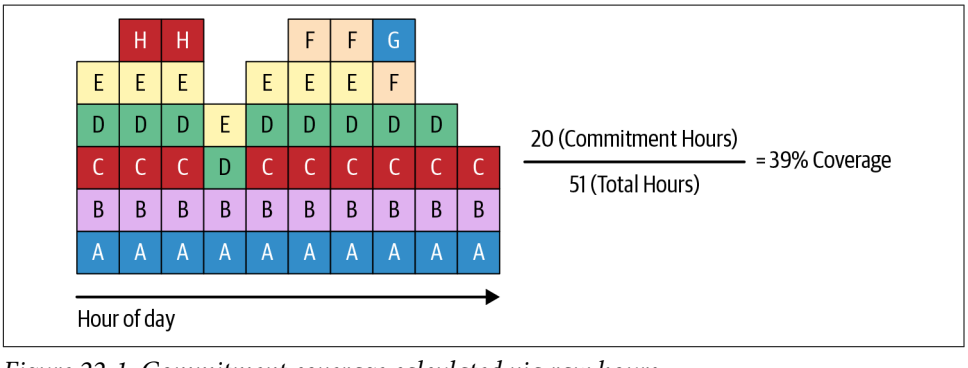


Figure 22-1. Commitment coverage calculated via raw hours

Even though the way coverage is measured in Figure 22-1 is correct, this may not be the right way to do it. With elastic infrastructure, resources come and go frequently, and there are some hours that shouldn't be covered by a commitment because there simply isn't enough usage during the period of time to warrant it. Ideally, you don't include all the usage that is under the break-even threshold, below which commitments would cost more than they save. By removing these noncoverable hours, you can measure coverage over coverable hours (see Figure 22-2).

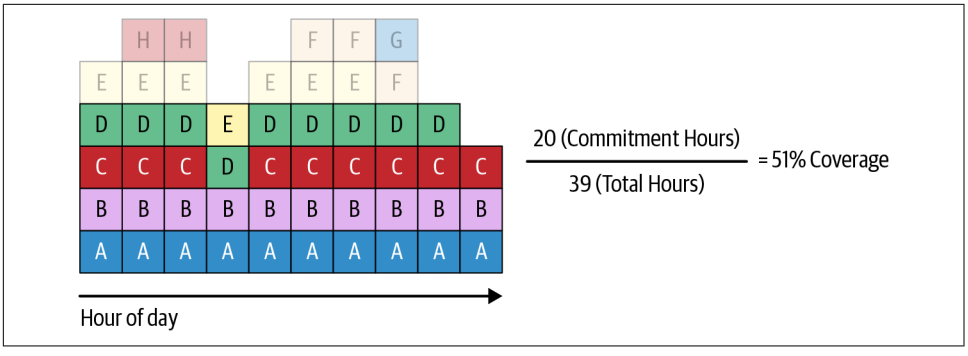


Figure 22-2. Commitment coverage calculated via coverable hours

If you remove noncoverable hours that shouldn't be covered at the peaks in the graph (those indicated as H, F, G, and E in Figure 22-2), you get a clearer picture of the true coverage percentage. A and B are covered by a commitment, leaving the hours covered by resources C, D, and E at on-demand rates. Buying an additional two commitments provides 100% coverage, and the two extra commitments would save money.

So far, you've been treating the cells in the graphs as if they are all the same thing: B is the same as A, which is the same as C. But in reality, each cell (or hour or second) may be a wildly different-sized usage. Usage marked as A may cost over $1 an hour, while B may cost $0.25 an hour. The amount you save on each commitment would then be vastly different. Primarily, you buy commitments for savings, so measuring them as consistent blocks—when they are so different—provides misleading results. The goal is to work out how much of your coverage is actually generating savings.

Converting the hour cell into possible savings by covering it with a commitment clearly shows the effect of a commitment on your cloud bill. Adjusting the graph by the relative savings of each usage type, you end up with a very different picture (see Figure 22-3).

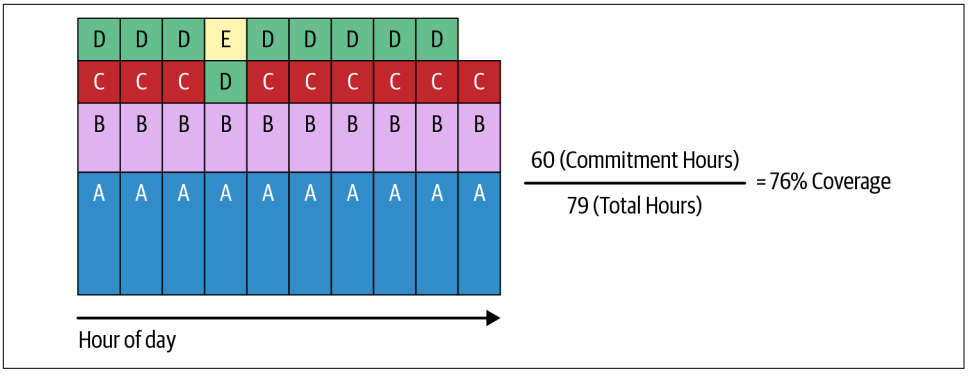


Figure 22-3. Commitment coverage calculated via weighted coverable hours

Figure 22-3 shows that you're actually closer to 76% of achievable savings. The original assessment in Figure 22-1 showed you as being in a very low-coverage situation. But the new weighted position, which also removes hours that shouldn't actually be committed, shows that your position isn't really that bad.

The key takeaway is that committing different resources realizes different amounts of savings. Not all usage is coverable, so including it all in committed coverage results in metrics that are more difficult to set targets against. However, by thinking in terms of achievable coverage, you can more accurately determine how and where to make improvements.

Savings metrics that make sense for all

Adding up the savings across everything that should be committed gives you a concept of *total potential savings*. For example, let's assume all of the cells in Figure 22-3 would save $102,000 per year when covered with a commitment (total potential savings). So, 76% of the $102,000 is the amount of savings currently being realized (realized savings), leaving the remaining 24% uncovered and at the on-demand rates (unrealized potential savings).

This connects back to the language of FinOps. When you move away from talking about usage hours and commitment coverage rates to a place where the conversation is "you are currently realizing 76% of your total potential savings of $102,000," this conveys to everyone in the organization where you stand and where you could go. Further, it removes the need for a deep understanding of the intricacies of commitment-based discounts and enables conversations within the organization. You've created a common lexicon for mutual understanding.

Combining metrics

To determine a performance metric for your commitments, you measure how much of a commitment is being used versus not being used over a specific time period. This is called your *commitment utilization*. At an individual commitment level you monitor by time, measuring the proportion of hours (or seconds) that a commitment applied a discount to your resource usage versus the number of hours (or seconds) you didn't have any usage for the commitment to discount.

However, if you roll your commitment utilization up to show one overall performance metric, the individual underperforming commitments can be hidden. Instead, you measure individual commitments and alert when they individually underperform. Let's look at an example of this in Figure 22-4.

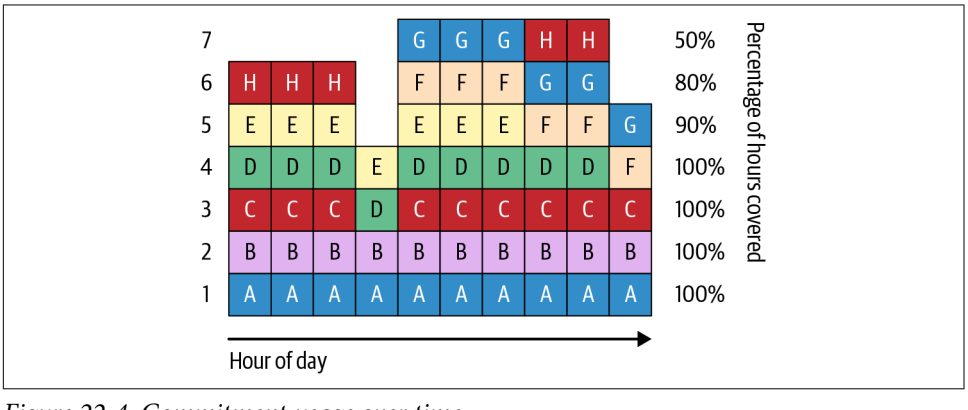


Figure 22-4. Commitment usage over time

You use the usage in Figure 22-4 to determine your commitment utilization, based on having seven commitments. You can see that the first four commitments are utilized 100%. The fifth commit will have 90% utilization and the sixth will have 80% utilization, with the last commitment at only 50%.

If you just average out the utilizations (100% + 100% + 100% + 100% + 90% + 80% + 50%) / 7, you get 88.5% utilized. An overall utilization rate of 88% on the surface appears pretty decent and wouldn't raise concerns. Had you set a target utilization of 80%, MDCO would reflect no changes needed. But remember: there's an individual commitment achieving only 50% utilization, and that deserves your attention and possible modification.

Data Driven

For years, teams have been operating their services using metrics to monitor items like service performance, resource usage, and request rates as an early warning

system, alerting the teams when thresholds are passed or events triggered and consequent adjustments are needed.

Operating FinOps optimizations without metrics provides no real method of identifying when optimizations are meeting goals. In fact, without metrics you can't even set goals. The tracked metrics build upon the feedback loop we discussed in Chapter 9. Metrics help you identify where you're unoptimized or underperforming so you can make improvements. In other words, metrics help you measure how much your cost optimizations are saving.

Applying MDCO to FinOps allows you to measure the desired outcomes of a cost optimization. When metrics show that a change has not been successful, you can use this as an early warning that the change needs further adjustment, or you can roll it back entirely.

In Episode 11 of the FinOpsPod, TJ Johnson from Box.com explained how most visualizations in FinOps help you answer four main questions that tie nicely with an MDCO approach, though he uses some slightly different terminology:

- *How am I doing so far?* Measurement of where you are currently.
- *How do I get better?* Identifying the optimizations you can make and helping you select which areas you would like to tackle.
- *Am I actually getting better?* Measuring how you are improving over time, forming a set of key momentum indicators (KMI), which help you monitor if you are getting better—or worse—over time.
- *Have I reached my goal?* Providing feedback on when you are completing the goals you have previously set.

Measurements that answer these four questions play into a DevOps mindset, where there's constant data flowing in to feed into the decision making of the application team, the finance team, and the business. Listen to the episode to get the full story of how he does it at Box.com.

Metric-Driven Versus Cadence-Driven Processes

In Chapter 18, we discussed strategies around commitment-based discounts. Most companies, when starting out with commitments, will purchase their commits on a schedule (monthly, quarterly, or even yearly). We call this *cadence-* or *schedule-driven cost optimization*.

If you purchase in slow cycles, you end up with large amounts of uncovered usage. Figure 22-5 shows some usage over time, and in dark gray you see the commitment coverage you have. You can see from the graph that purchases of commits are occurring on a relatively set cycle, and between these purchases resource usage continues to grow. By buying commits more frequently, you can achieve greater savings.

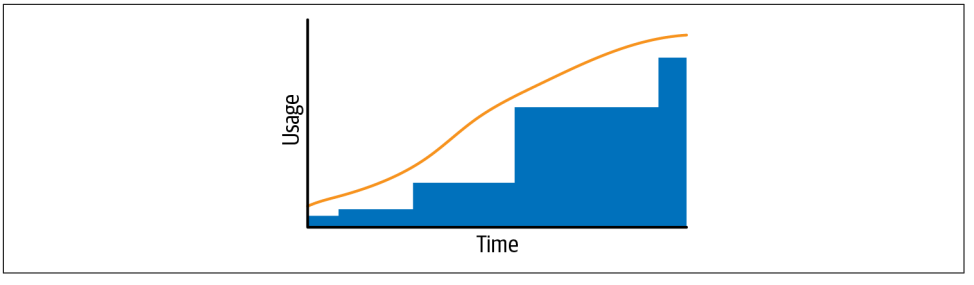


Figure 22-5. Commitment coverage when performed infrequently

Figure 22-6 shows a much more frequent commitment purchase program. The frequency isn't always the same over time, but the resulting commitment coverage follows the resource usage growth tightly.

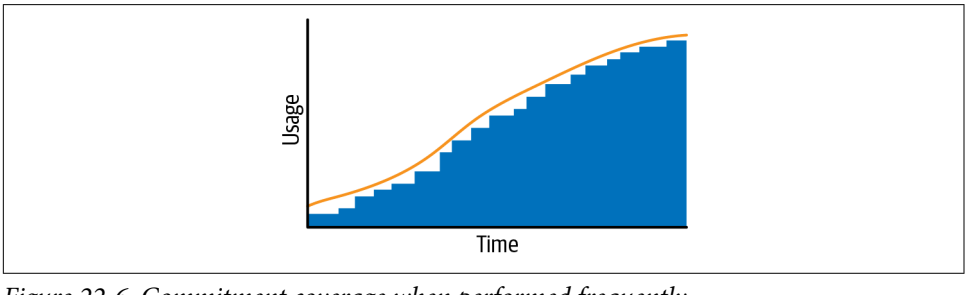


Figure 22-6. Commitment coverage when performed frequently

Schedule-driven cost optimization is fine to get started, but continuing to perform purchases slowly leaves savings on the table. However, looking at commitments too often will result in time wasted, so you need to strike a balance.

You can quickly determine when you're missing your goals by using metrics with set targets.

Metrics will trigger alerts to perform optimizations only when needed. MDCO can be used for the entire gamut of available optimizations. It's critical to measure metrics properly, so that optimizations are triggered at the right time. It's also important to set achievable targets.

Setting Targets

For some metrics, like commitment coverage, we recommend setting a conservative goal at first and then working your way to a high target such as 80%. A low target does not make sense, however, for utilization. Applying a "start low" model to commitment utilization would end up costing you more than it saves.

Stories from the Cloud—Mike

At Atlassian, my team aims for 90% utilization of a commitment to get the most possible savings. When utilization drops below 90% for an individual commitment, it triggers the computers to create a ticket, telling me that changes are required. At that point, modifications or exchanges of affected commitments drive the utilization levels back up. This process shows how a correctly measured and acted-upon metric promotes data-driven decision making.

Let's look at applying metrics to usage reduction methods. Removing unused resources and resizing underutilized resources also generates savings. The total potential savings as a percentage of cost is referred to as a *wastage percentage*. A team spending $100,000 a month with potential savings of $5,000 would be said to have 5% wastage. Alternatively, a team spending $10,000 a month and a savings potential of $2,000 would have 20% wastage. Teams can be compared when measured this way, and generalized targets can be set for wastage. When wastage exceeds a set amount, you can focus on reducing it.

Taking Action

We mentioned that MDCO encompasses all three phases of the FinOps lifecycle. We've shown how you can use this data to build visibility, measure potential optimizations, and set goals based on these numbers, all of which are inform and optimize phase tasks, yet we've placed this chapter in the operate phase part of the book. That's because the success of MDCO depends on what you do after you've set up reporting and alerting of MDCO and these alerts fire, requiring you to respond to needed changes in your optimization performance.

As previously mentioned, without clearly defined processes and workflows, it's unlikely you'll meet your organization's FinOps goals. It's important to specify who is responsible for actioning an MDCO alert, what the correct approval process is for things like buying new commitments, and what the correct order of operations is for modifying and/or exchanging existing commitments.

Have a preapproved commitment purchasing program within your organization. Giving the FinOps team the freedom to buy commitments as needed—up to a certain amount per month—reduces the lag time between a commitment coverage alert and the purchase of more commitments to cover the shortage. Improving processes helps reduce the delay in taking action.

There's a fine balance between proactive and reactive. What should be reactive? What should be proactive? Don't react to time as you would with cadence-driven cost optimization; aim to react to cost data as MDCO enables. Use this data to react faster. Capabilities like forecasting can help you predict when you will cross thresholds required before they happen, enabling you to be proactive. By ensuring this proactive data-driven prediction, you'll be ready to react quickly to real-world cost scenarios that can have dramatic effects on your cloud bill.

Bring It All Together

MDCO gives you the confidence that metrics will trigger you to adapt when needed. It moves you from reacting to an arbitrary schedule on the calendar (such as monthly RI management) to reacting to changes in data. But don't improvise your reaction; aim to have plans in place for the functional activities that you will do when the metric goes outside of the bounds, such as purchasing additional RI commitments. And be sure to plan for who will take which actions.

Ashley Hromatko, formerly Director of Cloud FinOps at Pearson, laid out this example—see Table 22-1—of how she used MDCO (coupled with assessment lenses from the FinOps Foundation's Finops Assessment) to define a clear process for who would determine changes needed to a cost threat, and completed the loop with automation to action changes in the environment, which resulted in significant cost savings.

Table 22-1. Policy changes and automation processes put into place to manage multipart uploads

Phase	Lens	Process	Action
Inform	Knowledge	Awareness of impact for change	800+ AWS accounts and no lifecycle rule configured for S3 incomplete multipart uploads
Inform	Process	Who will make decisions	Spending 11K per month on incomplete multipart uploads, no single account is exceeding more than 1K
Optimize	Metrics	Baseline KPIs for success	FinOps submit policy to Cloud Governance board for review, vote to move to implement
Optimize	Adoption	How will implementation be communicated	Decision made for remediation after 14 days and ability to opt-out
Operate	Automation	Offload repetitive tasks to tooling	Cloud Custodian policy created by Cloud Management engineering team

The outcome was a process that attached a lifecycle policy to all S3 buckets and automated the removal of incomplete multipart uploads after 14 days. The policy resulted in significant automated cost savings the next month, as shown in Figure 22-7.

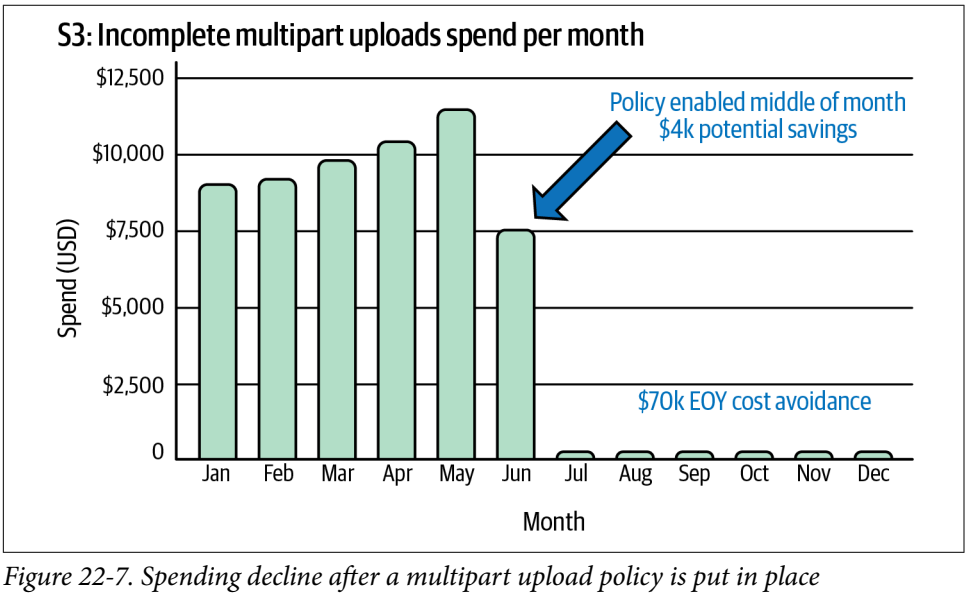


Figure 22-7. Spending decline after a multipart upload policy is put in place

Conclusion

Metric-driven cost optimization encompasses all three phases of the FinOps lifecycle, with visibility and targets driving the cost optimization process. Automation plays a major role in providing a repeatable and consistent process.

To summarize:

- All cost optimizations are driven by metrics, and all metrics require achievable targets.
- Computers should perform repeated data analytics and alert when metrics are off-target.
- All changes being performed should be reflected in the metrics. Ultimately savings should increase, and cost optimization efficiency should remain high.
- MDCO is about knowing exactly when is the optimal time to perform optimization processes.

Just when we believe we have this FinOps world under control, engineers innovate and introduce a whole new world on top of the cloud using containerization. Next, we look into the impacts of clusterization using containers and how you can use your existing FinOps knowledge to solve the new problems this creates.

CHAPTER 23

FinOps for the Container World

To support the adoption of microservice architectures, containers have gained popularity. Over the last few years, the number of container environments being run by organizations has rapidly increased. Managing a single running container instance is quite simple, but running hundreds, thousands, or tens of thousands of containers across many server instances becomes difficult. Thus, along came orchestration solutions like Kubernetes, which enable DevOps teams to maintain the configuration and orchestrate the deployment and management of fleets of containers.

Since containers and container orchestrators are becoming a popular choice for many teams, it's vital to understand the fundamental impact of these containerized workloads on FinOps practices.

The most important reason this impacts FinOps so much is that most container environments are complex shared environments. Shared resources like containers—which run on shared cloud compute instances—cause challenges with cost allocation, cost visibility, and resource optimization. In the containerized world, traditional FinOps cost allocation doesn't work the way it does with just virtual machines (VMs). You can't simply allocate the cost of a resource to a tag or label, because each cloud or on-premises resource may be running a constantly shifting set of multiple containers, each supporting a different application. They also may be attached to different cost centers around the organization. Not to worry, though. In this chapter we'll look at changes and adjustments that will allow you to successfully operate your FinOps practice in the container world.

Containers 101

Let's quickly run through the basics for anyone not familiar with containers. It will also be helpful to create a common understanding of these components throughout this chapter.

Containers are, quite simply, a way to package software. All of the requirements and settings are baked into a deployable image. Container orchestration tools like Kubernetes help engineers deploy containers to servers in a manageable and maintainable way.

There are a few key terms we will use throughout this chapter:

Image
: A snapshot of a container with the software that needs to be run.

Container
: An instance of a container image; you can have multiple copies of the same image running at the same time.

Server instance/node
: A server (e.g., EC2 instance, VM, physical server).

Pod
: This is a Kubernetes concept. A pod consists of at least one container but can also contain a group of containers and treats them as a single block of resources that can be scheduled and scaled on the cluster.

Container orchestration
: An orchestrator manages the cluster of server instances and also maintains the lifecycle of containers/pods. Part of the container orchestrator is the scheduler, which schedules a container/pod to run on a server instance. Examples include Kubernetes or AWS Elastic Container Service (ECS).

Cluster
: A group of server instances, managed by container orchestration.

Namespace
: Another Kubernetes concept, a namespace is a mechanism for isolating groups of resources within a single cluster where pods/containers can be deployed separately from other namespaces.

A Kubernetes cluster (see Figure 23-1) consists of a number of nodes (server instances) that run containers inside pods. Each pod can be made up of a varying number of containers. The nodes themselves support namespaces used to isolate groups of pods.

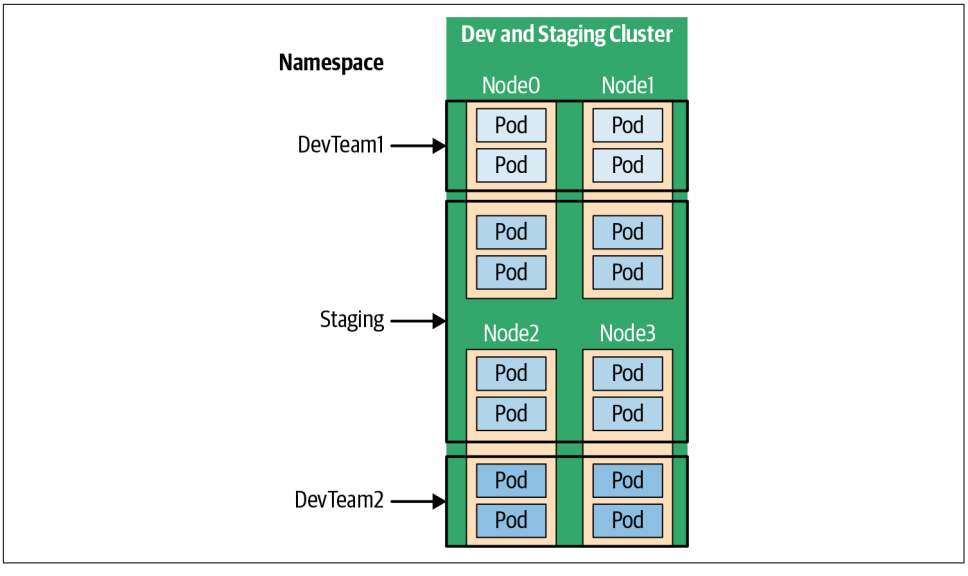


Figure 23-1. Basic view of a Kubernetes cluster

The Move to Container Orchestration

Engineers make the decision to move to containers for a multitude of reasons including availability, reliability, scalability, and ease of management. Remember, FinOps is about making sure that you are doing what's best for the business, continuously balancing between cost and technical needs. *Cost is not the only driver for the migration to containers,* but having the ability to cost out an application in a container model and a VM model (with all attendant costs and implications on both sides) is helpful context to provide the business.

When you change from a large number of individual cloud server instances to a cluster orchestration solution like Kubernetes, you usually end up with a smaller number of larger servers. By packing multiple container workloads into these larger servers (also known as *bin packing*), you can gain efficiency through better resource utilization by running many containers isolated on the same node.

An analogy might help you picture this idea: each server instance in a cluster is like a game of Tetris. Containers are the blocks of different shapes and sizes, and the container orchestrator is trying to place as many of these "blocks" (containers) into the servers as possible.

Platforms like Kubernetes allow you to set different resource guarantees, allowing you to pack more containers onto a server instance, which we'll cover in more depth later in the chapter.

You lose visibility into the individual costs of each container when running multiple containers on a single server instance, since the cloud service provider will provide billing data only for the usage of the underlying server instance. When a server instance is running multiple containers, you need more information in order to divide up the usage costs.

In other words, each cloud resource that's shared by multiple workloads needs to have supplemental data, so it can show how much of the cloud resource was used by each workload. Containers do not consume servers equally. Some containers can be configured to use more of a server instance than others. Some containers run for weeks or months. But research shows that most containers often run for short periods, less than 12 hours on average, and some might only run for a few seconds. With different sized containers coming and going at different rates, taking infrequent snapshots of your cluster usage won't provide sampling sufficient for the detailed cost allocation you need. Detailed data that tracks all the containers as they come and go will help you identify how the individual server costs are to be divided, and how much of the server was used by a container for service A versus a container for service B. You will see what this data looks like in "Container Proportions" on page 357.

One of the difficulties collecting this data, though, is the sheer volume of it. The cloud billing data files are already extremely large, but you may have many times the number of containers as VMs, and each one may shift around from resource to resource many times, creating a new line of data each time. It is a difficult challenge that requires some specialized tooling if you aim to solve it in a detailed and granular way.

FinOps practitioners need to prepare teams for the impact of shared resources on cost visibility and cost allocation processes. It's vital to have a clear understanding of the requirements needed to maintain successful FinOps practices in the container world, and then to be sure they get implemented by the DevOps team.

The Container FinOps Lifecycle

When you look at the challenges that containerization poses for FinOps—cost visibility, cost showback/chargeback, and cost optimization—you quickly realize that you're encountering the same difficulties you faced as you moved into the cloud. Containers represent another layer of virtualization, also called containerization, on top of cloud virtualization. While this might feel like you have moved backward, keep in mind that you have already accumulated all the knowledge, processes, and practices necessary to solve these challenges. You simply need to base your approach on how you solved for the cloud in the first place.

Therefore, you can divide the containerization FinOps challenges into the same inform, optimize, and operate lifecycle that you applied to the broader cloud FinOps.

Container Inform Phase

Your first focus should be to generate reporting that enables you to determine the cost of individual containers on the clusters that teams are operating. In other words, you need to build visibility into your container spending.

Cost Allocation

You will be charged by a cloud service provider for every server instance that makes up a cluster. When containers are deployed into a cluster, they consume some amount of the cluster's resource capacity, such as CPU, memory, and storage. Even if the processes inside the container don't consume all of what was provisioned for that container, in some cases it blocks other containers from using this capacity, and therefore you have to pay for it. It's just like when you provision a server with your cloud service provider. You pay for that server resource, whether you use it or not. The network traffic costs from and to your services is a challenge on its own and should be included as well. Network costs attributed to a single node or server instance may be supporting a number of containers, and the related costs will need to be split up by each individual container's usage. Network becomes a form of shared costs that require an equitable distribution model, ideally by evaluating the actual network traffic volume of each container and allocating the network costs based on these amounts.

To allocate the individual costs of a container that runs on a cluster, you'll need some way to determine how much of the underlying server the individual container consumed. We'll cover this more in the next section.

You also need to take into account the satellite costs of a running cluster. Management nodes, data stores used to track cluster states, software licensing, backups, and disaster recovery costs are part of your cost allocation strategies. These overhead costs are all part of running clusters and must be taken into account in your cost allocation strategy.

Container Proportions

The biggest issue with governing clusters is that you're sharing underlying resources, so you can't get complete visibility into allocation by using your cloud bill alone. Remember, a cloud service provider can only observe metrics provided by the VM hypervisors, not operating systems or processes, unless you're using their managed service offerings. This lack of visibility means providers are unable to track your containers and how much of the server they use. So for your FinOps team to divide

the underlying server costs out to the right teams and services, you need the teams who run the container clusters to report on which containers are running on each server instance. By recording the proportion each container is using of each of your servers, you can enrich your cloud billing data.

Custom container proportions

Measuring the proportion consumed by each container of the underlying cloud server instance is a complex process. But you can start by using the amount of vCPU and memory used.

Table 23-1 shows an example to help you understand this data.

In the table, there's a server instance with four vCPU and 8 GB memory that costs $1 per hour. This server is part of a cluster, and during a particular hour runs containers 1–4.

Table 23-1. Example container allocation on a server instance

ServerID	Container ID	Service label	vCPU	Mem (GB)	Time (mins)	Proportion
i-123456	1	A	1	2	60	25%
i-123456	2	A	1	2	60	25%
i-123456	3	B	0.5	1	60	12.5%
i-123456	4	C	1	2	15	6.25%

The proportion calculation appears pretty simple when you look at containers 1 and 2. They use a quarter of the vCPU and memory for the whole billing hour, so you allocate a quarter of the server costs. As both of these containers are running for the same service (A), you allocate 50% of the server costs to service A.

Container 3 appears to have a similar story. However, this time it's using even less of the server.

When you get to container 4, you see that it's using 25% of the server instance, but this time only for 15 minutes of the hour (25%). That reduces its proportion data when you multiply it by the time: $0.25 \times 0.25 = 6.25\%$.

There are cases where the proportion of metrics isn't neatly symmetrical to the underlying server, which causes problems when you are calculating the proportions. We've seen two main approaches for solving this issue:

Most impactful metric
: If a container uses more vCPU than memory, that ratio is used to calculate the proportion.

Weighted metrics
: Weight the metrics (vCPU, memory, etc.) first and then use them to calculate the proportions.

While we recommend starting with vCPU and memory, they're not the only metrics that could be important when you're determining the proportions for your containers. You may need to take into account other metrics, such as local disk usage, GPU, and network bandwidth. Depending on the types of workloads scheduled onto a cluster, the overall metric weighting that is most relevant to specific clusters could be different. It is also important to note that with Kubernetes there is a difference between requested resources and consumed. Often a pod will use fewer resources than it requests, but the opposite can also occur. When measuring the proportions of usage, we recommend you use the greater of the requested resources and the consumed resources by the workload.

Even if you allocate the costs of your clusters in the previous manner, you can still end up with unallocated capacity costs. It's unlikely your clusters will be 100% used—or requested—all of the time by the running containers, which leaves excess capacity that has not been allocated—nor is it being used.

Figure 23-2 shows a single server instance that has an hourly cost of $40. On this node there are two running containers. Team A and B get allocated $15 and $11 of the costs, respectively, even though some amount of their requested capacity within their container is unused and could be considered waste. There is still $14 of spare node capacity (allocatable capacity) on the server instance that needs to be assigned to someone's budget.

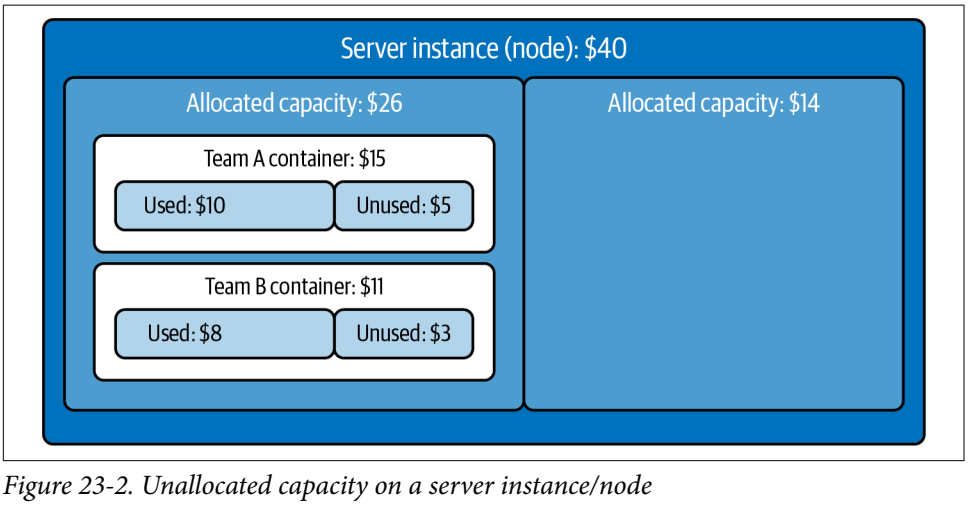


Figure 23-2. Unallocated capacity on a server instance/node

We have not yet seen a standard approach to allocating the excess, allocatable node capacity. One option is to have the team managing the server instance itself cover the cost; another is to spread the cost of the allocatable node capacity equitably between all the containers within the node based on their respective proportion of usage of the overall node.

Reserved capacity is a soft limit, and pods are able to exceed its set limits if there is additional allocatable capacity remaining within the node. As such, the amount of capacity that a pod consumes can be more than 100% of its reserved capacity. This means that the pod's portion of the overall node's costs would increase during the period of use, and less of the unreserved node capacity would be left unused.

This leaves the question of who should pay for the excess capacity: the team managing the cluster or the teams who own the workloads running on the server instances?

Having spent time with organizations that are successfully allocating their container costs, we've identified two main strategies for assigning these unallocated costs when they exist:

Assigning the idle costs to a central team (recommended option)
: Since you need an allocation strategy to make sure that your cluster costs can be 100% allocated, some organizations take those excess capacity costs and assign them to the team that runs the cluster. These teams have their budget and forecasts on cluster costs, so by mapping the excess capacity costs to them, you incentivize the group to optimize the container scheduling, reduce the size of the cluster as permitted by the workload, and keep cluster waste to a minimum, thereby reducing those excess capacity costs.

> *The team running and administering the cluster gets visibility into the allocatable costs on cluster/node level. In most cases the actual usage through the deployed services/pods is dependent on the resource requests that are set by the actual users like the platform team running the platform services for everyone and the developer teams with their respective applications. The platform team is positioned best to rightsize the cluster, switch to other instance types, or set up a cluster autoscaler to overall reduce the allocatable costs by better sizing the cluster, which increases efficient utilization of the cluster's resources.*
>
> —David Sterz, Mindcurv, Container Cost Allocation Working Group lead

Distributing the idle costs to the teams using the cluster (alternative option)
: The second strategy is to distribute the excess capacity costs to the teams running containers on the cluster. You determine what gets distributed, and where, by using the same proportion data you use to allocate the costs of the containers.

This method avoids the need for a central catch-all budget for the excess capacity costs.

Before deciding on an approach for allocating the excess capacity costs, we recommend you engage both the teams running the clusters and the teams consuming the clusters, so you can reach agreement on how to proceed.

Container proportions in Google Cloud

When you use custom machine types, Google Cloud charges for vCPU and memory separately, so if these are the metrics you're going to use for container cost allocation, you don't need to determine the proportion of the underlying server consumed by a container. If you record just the vCPU and memory allocated to each container that runs on your cluster, you can use the normal Google Cloud pricing table to allocate charges to each container. Your total cluster costs will be represented as an overall vCPU and memory charge. When you subtract the recorded container costs from these values, you are left with the unused cluster costs.

When you are using predefined machine types, however, you will need to record custom proportion data for each container, as previously discussed.

Tags, Labels, and Namespaces

Like server instances, containers without any identifiers make it hard from the outside to determine what is what. Realistically, you could end up with hundreds or even thousands of small containers. Having a tagging/labeling strategy will enable you to categorize your usage. Remember, you deal with containers in the same FinOps pattern as native cloud, which we hope you are getting used to by now. So it's not surprising that your standard cloud tagging strategy will apply nicely to containers.

In the container world, you can tag/label your containers. These tags form the identification that you use to allocate costs. Namespaces allow you to group resources within a single cluster. Creating namespaces for teams, or for specific services, allows you to create a coarse-grained cost allocation group.

Containers or pods themselves can also be labeled when they are launched, providing a more fine-grained cost allocation mechanism if needed, though the ability to see these labels and divide cost with them is done with tools outside the cloud billing data.

Container Optimize Phase

Like the inform phase, the optimize phase in the container world is a direct adaptation of your original FinOps practices built for cloud platforms.

At first glance, it might seem that containerization solves the rightsizing problem. After all, having more things running on the same server instance appears more cost-effective. But as you've seen so far, it's a complex task to measure whether or not you are getting savings from your clusters.

We have heard FinOps practitioners state, "Now that you've moved to containers, you don't need to rightsize." Or "Idle resources are no longer an issue." While there are some elements of truth to these statements, they don't tell the full story. Let's take a look at how your FinOps practices have to evolve in order to be successful.

Cluster Placement

If you run all of your containers within one cluster, you may not be able to optimize the cluster for different environments. Running development containers on spot instances, or in a region that is closer to developers, is easier when they run on a development-specific cluster. And further, it helps if you run this development cluster in a development account separate from production.

The alternative is to run multiple clusters, but running too many clusters can be hard to manage and isn't cost-efficient. Each cluster will have management overhead, both in terms of cloud resources and staff time needs. In addition, it's harder to maximize cluster utilization when you have workloads spread out over many clusters. The more containers are spread across clusters, the less chance containers will fill your running servers, which means lower utilization.

Finding a balance here is just as important as finding the balance of accounts for the cloud resources that we discussed previously. Working alongside the teams that are managing and using your clusters is the best method of finding the happy medium between smaller domain-specific clusters and larger general clusters within your organization.

Container Usage Optimization

Now we'll work through adapting the usage optimizations discussed in a broader optimize phase of the FinOps lifecycle.

There is a shared responsibility model for optimizing containers: centralized teams running the cluster are responsible for the sizing of the cluster, including horizontal/vertical rightsizing and workload placement, whereas teams running containers within the cluster are responsible for setting and using the appropriate amount of requested or guaranteed capacity.

Centralize the rightsizing of clusters

The responsibility for the optimizations in this section typically falls to the centralized teams managing the clusters and server instances.

If you find that your clusters aren't cost-efficient, the most likely causes are poor scheduling and underutilized cluster resources. Cluster instances that remain running after all running containers have stopped, or only one or a few containers running on a large cluster instance, will mean that the instance is largely underutilized—leading to higher costs per cluster and a need to rightsize.

In the container world, there are multiple strategies that need to be applied by your central cluster management team for rightsizing:

Horizontal rightsizing
: This refers to how well you pack your containers onto your server instances. The tighter things are packed together, the fewer server instances you need to run at a given point in time. When this is done correctly, your clusters can scale in the number of instances needed while realizing cost savings by avoiding running server instances that you don't need. Container orchestrators should be looking for opportunities to relocate containers to increase utilization of server instances while freeing up server instances that can be removed from the cluster—using autoscaling—when the load drops. Tracking the amount of unused cluster capacity over time allows you to measure changes in the scheduling engines. If you then update/tune the scheduler so that it manages to pack your containers onto fewer server instances, you can measure the increased efficiency.

Vertical rightsizing
: This refers to changing the size of server instances. While you should expect a container orchestrator to pack your containers onto as few servers as possible, there are other factors to consider. For high availability, some containers should be running on different underlying server instances. If this leaves your server instances underutilized, resizing the server instances to use less vCPU and memory will lower costs.

Workload matching
: If your container workloads aren't matching your vCPU-to-memory ratio (using more memory than vCPU, for example), then there's no memory for a container to use when there is free server vCPU. When your data tells you that your server's

balance between vCPU and memory doesn't match your container workload, you can run a server instance type that is a better match (e.g., one with more memory per vCPU) at a reduced hourly rate, thereby lowering costs. It is possible to use a mix of server configurations within the cluster and assign workloads to the servers that best match the container workload.

Decentralize the rightsizing of containers/pods

Teams running containers within a pod are responsible for correctly sizing their requests of capacity from the clusters.

Even within containers that have been efficiently bin packed, there's a risk that individual containers scheduled onto your clusters will be inactive. As you now know, containers consume a portion of cluster resources, so if a container is larger than needed, it will be less cost-efficient.

Instead of tracking server utilization metrics looking for idle server instances, teams are now required to track container utilization metrics to identify unused container instances. Again, similar concepts used to address idle server instances apply to containers.

Cluster orchestration solutions like Kubernetes allow you to request different resource class allocations on the scheduled containers called quality of service (QoS) classes. Each QoS class brings with it different models for allocating capacity and affecting the related allocation of costs:

Guaranteed resource allocation
: For critical service containers, you might use guaranteed resource allocation to ensure that a set amount of vCPU and memory is available to the pod at all times. You can think of guaranteed resource allocation as reserved capacity. The size and shape of the container do not change. This class has the highest potential for idle costs for containers when set to high and the actual usage is low.

Burstable resource allocation
: Spiky workloads can benefit from having access to more resources only when required, letting the pod use more resources than initially requested when the capacity is available on the underlying server instance. Burstable resource allocation is more like the burstable server instances offered by some cloud service providers (burstable performance instances like the T3 from AWS or the F4 from Google Cloud), which give you a base level of performance but allow the pod to burst when needed. While this is an easy way to start, more cost-efficient with a low request on resources but no limits, it can also lead to spiky unforeseeable resource consumption that can harm other workloads or a decrease in service performance due to the lack of available resources.

Best-effort resource allocation
: Additionally, development/test containers can use best-effort resource allocation, which allows the pod to run while there is excess capacity but stops it when there isn't. This class is similar to preemptible VMs in Google Cloud or spot instances in AWS. From a purely cost-efficient point of view, that class is the best as it has zero idle costs; however, there is absolutely no guarantee resources will be available at any time, and it should be used with caution for only specific workloads on production systems.

When the container orchestrator allocates a mix of pods with different resource allocation guarantees onto each server instance, you get higher server instance utilization. You can allocate a base amount of resources to fixed resource pods, along with some burstable pods that may use up the remainder of the server resources, and some best-effort pods to use up any spare capacity that becomes available.

Server Instance Rate Optimization

Unless you are using serverless clusters—which we will look into next—your clusters are made up of regular cloud server instances.

If your clusters run a lot of best-effort or restartable pods, there's the option to run part of the cluster with *spot instances*. Spot instances let you take advantage of possibly massive savings from your cloud service provider. But running them inside a cluster requires you to consider the added risk that your server instances—and the pods they're running—could be stopped by the cloud service provider with little notice.

These server instances can be requested, just like any other instance. Just because you run cluster orchestration software like Kubernetes doesn't mean you should exclude your clusters from normal reservation/commitment programs. Often, we see containerization lead to more stable cloud server instances. Containers can come and go without changing your cloud servers overall. This can mean simpler reservation planning and overall better reservation utilization. Reserving the instances that are running all the time can lead to savings well over 50%, as we discussed in Chapter 17.

Container Operate Phase

As you go through the broader FinOps operate phase in this part of the book, you should consider how many of the operations that you perform at a cloud resource level can be applied to the container world. Scheduling development containers to be turned on and off around business hours, finding and removing idle containers, and maintaining container labels/tags by enforcement are just some of the one-to-one parallels you can draw from the broader FinOps operate phase.

Serverless Containers

Serverless containers, offered by cloud service providers like Fargate for AWS, add an interesting twist to your FinOps strategies around containers. With self-managed clusters you need to consider the cost impact of running more clusters. There's an added cost and maintenance requirement of management compute nodes for each cluster, which can lead engineering teams to opt for fewer clusters in centralized accounts.

With the cluster management layer being managed by the cloud service provider and the compute nodes being abstracted away from you, there's less need to centralize your workloads. In the serverless world, tracking the efficiency of the scheduler and measuring how well your containers are being packed onto the cluster instances isn't an issue. This responsibility has been offloaded to the cloud service provider. Monitoring for overprovisioned containers is still a requirement, however, as today's offerings for serverless containers allow only certain sizes of vCPU and memory combinations.

Cost allocation is done for you, as each container task is charged directly by the cloud service provider. This is because they can now see what containers are running on the servers. Therefore, the challenge of proportioning costs and dividing shared resources is removed. When a container is running on a server instance inside your account, removing the container creates free space on the cluster but doesn't reduce costs until the compute node is removed from the cluster altogether. With serverless containers, when they're removed, you stop being charged.

This may sound like good news, but as with everything in FinOps, there are multiple considerations. For example, having the cloud service provider manage the cluster means there are fewer configuration items that your teams can tune, which might keep you from optimizing the cluster for your workloads.

Conclusion

The problems with containers are similar to the issues you initially face when moving into the cloud, so you can use the existing FinOps practices to solve these issues.

To summarize:

- Containerization won't let you avoid FinOps, given that FinOps principles still apply and that the need to manage costs is just as important for containerized applications.
- Unless you're using the cloud service–managed containerization platforms, you will need to gather supplemental data about how your servers are used by the running containers.

- Tracking the proportions of server instances your containers consume and pairing that information with your cloud bill enables true cost allocation.
- Containerization does not necessarily mean efficient: monitoring server usage and performing rightsizing and scheduler optimizations will be required.
- Serverless orchestration reduces the FinOps overhead of allocating costs, but comes with a high potential for increased costs.

The next chapter digs into the process of building partnerships with your engineering teams.

CHAPTER 24

Partnering with Engineers to Enable FinOps

FinOps is a team sport, and very little gets done without the engineers. In this chapter, we're going to talk about how to instill a culture of FinOps by building up engineering motivation to be cost-efficient rather than dictating a mandate. Mandates only provide short-term action while they are top of mind for leadership. Ultimately, the pressure from mandates lessens and things go back to business as normal, with cost inefficiency creeping back in.

In this chapter, we'll look at why a software engineer's day-to-day life makes considering cost optimization difficult and we'll cover the transitions that engineers tend to make as they begin to integrate cost considerations into their daily workflows.

Many engineers have come from a world where cost considerations weren't historically part of their job. Now, in the cloud, they are being asked to consider cost as a new constraint. Conveying the context of why the cost of variable infrastructure is now an important piece of their job and its impact on the business is key; otherwise, it may get deprioritized.

In a mature practice, engineers and developers are at the pointy end of the FinOps spear and their early involvement, alignment, and buy-in are crucial to success.

Integrating Us with Them

When revisiting the first edition of this book, we noticed how much it used an *us versus them* tone when it came to FinOps teams and engineering teams. Some of the phrasing implied that engineers were somehow outside of the work of FinOps and that they didn't generally care about cost. This is far from the truth in companies who are mature in their cloud—and FinOps—transformation. Lightly engaged engineers who only do reactive cost optimization reflect an immature FinOps practice and maturity, where inefficiencies frequently are baked into the development process and

FinOps processes are limited to cleaning up the waste after the fact reactively. Instead, mature practices partner with engineers early on in service and architecture design to proactively avoid inefficiencies in the way services are deployed on cloud.

The personas driving FinOps have shifted in the last few years. In the State of FinOps 2022, the team responsible for driving FinOps overwhelmingly reports to the CTO or CIO and is staffed with—and increasingly led by—engineers. Once the cost-conscious culture is instilled, the work of FinOps itself is performed by engineers around the organization, leaving more focus on other functions for the central FinOps team. As we mentioned in Chapter 15, many critical functions of FinOps—like usage reduction and optimization—are not typically done centrally by the FinOps team; rather, those responsibilities are distributed directly to engineering teams in various business units or groups. The engineers who are deploying services know their infrastructure better than anyone and are best suited to optimize it.

What's on the Mind of the Engineer?

Engineering motivations stand in contrast to finance team motivations. The latter tend to focus on business viability, gross margins, and future financial stability of the business through the lens of dollars spent against budgets and forecasts. Engineers are motivated to innovate, deliver value through software improvements, and generally make things more efficient across a range of areas including security, performance, uptime, and reliability.

In Chapter 3, we talked about how to increase the motivation of teams and reduce the loss of trust. Recall the *action scale* introduced, where there is balance between the motivation and the effort required. There is sometimes a misperception that engineers aren't motivated to focus on cost efficiency; in reality, their FinOps work simply competes with other critical constraints and considerations.

To balance the action scale, FinOps practitioners need to work with engineers (and engineering leadership, who set their goals and incentives) to make cost efficiency as big a consideration as security, performance, reliability, or sustainability (as discussed in Chapter 19). Focusing on only one measure of success is not an option.

A perceived lack of engagement from engineers is not typically a lack of motivation. Due to the long list of areas for which they are responsible, they must make regular trade-off decisions. If FinOps requires a large—or unclear—amount of time, then it may not fit into the engineering schedule unless prioritized by leadership.

Stories from the Cloud—Abuna Demoz

Abuna Demoz, a Software Engineer who has led large engineering teams at Google Cloud and Amazon Web Services, shared the following insights:

> You should assume that most engineers start out wanting to complete as many tasks in the software lifecycle as possible ranging across development, documentation, monitoring, alerting, refactoring, scaling, etc. But then they hit the constraints of what can actually get done in a workday or by the target launch date for a new feature. This is especially true for senior engineers, where the demands are the highest. They are forced to constantly prioritize and make tough choices amongst all the tasks they need to do in timeframes that usually don't allow for anything outside critical requirements.
>
> This is where an organization's reward structure can come into play. Most engineers are going to prioritize according to what will either be rewarded (pay, bonus, etc.) or recognized (praise, leadership visibility, promotion, etc.). Using rewards and recognition to drive behavior can be very effective, but we must also understand the implicit trade-offs created therein.
>
> For example, in most organizations, engineers are rewarded for launching new features that customers use and ideally pay for. That means that when time is scarce it's easy to underinvest in reliability, maintainability, scalability, and especially cost.
>
> As organizations make the shift to FinOps, tying cost metrics and goals to engineer rewards is a great way to get everyone to pay attention to the same problems. Rewards can be tied to increasing tag coverage, reducing waste and underutilization, and/or improving forecast accuracy. Leaders' recognition of their team's work toward these goals helps to reinforce their importance.

It's important for FinOps to look for ways to reduce the load on engineers, especially with automation. As discussed in Chapter 8, putting FinOps *data in the path of the engineer* will enable faster decision making and reduce cognitive load lost to context switching. We don't want engineers to spend high amounts of time (or brain power) to figure out how their actions are affecting spend projections, not only taking them away from other priorities but also reducing the likelihood that they will tend to their FinOps responsibilities. FinOps reports should give engineers a clear outline of what actions they need to consider and the precise data supporting those actions, decreasing the load and time commitment needed for engineers to consider cost.

Constraints and the Solving of Hard Problems

Humans don't like to be constrained. But constraints are required for innovation.

> *To create something great requires two things, a plan and not quite enough time.*
> —L. Bernstein

Cost guardrails can often be met with a sentiment of "We went to the cloud to have fewer constraints, and now you are adding more constraints back on me." A further refrain we hear from engineering teams is "Tell me where you want to go, but don't tell me how to get there." By collaborating to set mutually agreed-upon cost guardrails in the form of budgets and forecasts—and providing trusted data on cloud spend and usage to teams managing infrastructure—FinOps can provide engineers the constraints needed to tell them *where* they need to go in terms of cost, while leaving to them the innovation of *how* to achieve that cost efficiency. It also helps if leadership provides the *why* of their cost efficiency work to the larger business.

A clear vision for the desired outcome delivered via a clear set of constraints is the goal. Without constraints, the development process can be too freeform and may not ultimately align with the desired outcome. Conversely, too many constraints means there is not enough room for creative engineering to occur, as it leaves no room for innovation. Some of the best solutions to cloud efficiency come by allowing room for engineers to be innovative, with clear cost constraints but not overly prescriptive guidance for achieving them.

Rather than allowing engineers to choose any available cloud architecture and then cleaning up associated cost inefficiencies later, early cost constraints reduce unnecessary waste and the need to clean up future expensive technical—and cost—debt. However, even an architecture designed to be cost-efficient at the start can almost always be optimized over time, as code efficiencies are introduced, new cloud features are released, and real-world load evolves.

Engineers like to solve hard problems. This is evident from the number of engineering success stories you hear at major cloud providers' annual conferences. Most of these talks cover how companies have used the cloud to scale, be more innovative, add new features, and introduce completely new business opportunities to an organization.

> *Engineers who build cloud solutions that fit within the boundaries of cost constraints should be celebrated as innovative. Give engineers a platform to share the technical difficulties they overcome with creative engineering solutions to meet their cost constraints.*
> —Benjamin Coles, Engineering Manager in FinOps at Apple

Chapter 13 explored different levels of cloud budgeting and how some teams new to cloud can push back on the concept of having a budget. Ironically, there has always been a budget for IT spending in most organizations with related hardware

constraints, but that spending was abstracted away via hardware that was purchased sometimes years in advance. Dollars themselves are being introduced as the new constraints, rather than quantities of available computing power as was historically the case.

Are You Building a Porsche or a Toyota?

Innovation can happen at high price and quality, or it can happen at lower price and lower quality. In your own team, consider whether you are building the Porsche of clouds or the Toyota of clouds. Either approach is valid. Both the Porsche engineer and the Toyota engineer have similar training but are presented with different sets of constraints. They both must constantly make a cost versus value trade-off, but under those different constraints. You don't have to be building the Porsche of clouds in order to be innovative.

The friction comes when one team (or team member) thinks they are following the Porsche model, whereas in reality the others are following the Toyota one. There's a gradient between optimum cloud and one that uses every feature possible at great expense.

A lot of people think that you need to be building the Porsche of clouds for engineers to be doing the best work. They design it to be the most available, the most fault tolerant, the most performant. The engineers who build the Toyota of cloud build highly efficient services that operate within tight constraints and can, in some cases, actually be solving a more difficult engineering problem than those Porsche engineers who can use any service available to them. Every time the budget gets squeezed, it introduces an opportunity to look at creative ways to optimize, such as spot, serverless, or new system designs to fit within those constraints. It's all still innovative engineering, just based on a different balance of the Iron Triangle of good, fast, cheap.

Adapted from Gabe Hege's talk at FinOps X. Gabe is a Staff Software Engineer in Cupertino.

Principles for Enabling Cost-Efficient Engineering

After introducing the concept of dollars as a constraint and the Toyota versus Porsche analogy in his talk at FinOps X 2022, Gabe Hege shared a set of principles for both business and technical people to remember when working on a FinOps transformation. Gabe is an engineer who works in a FinOps enablement team at a Fortune 100 company, but who reports to a finance director who ultimately reports to the CFO.

As a cautionary tale, Gabe quipped that the "only limitation of cloud is how deep your pockets are. But there's a big difference between limitless and free. Cheap used a lot is expensive."

The following sections summarize Gabe's six principles for FinOps teams to work more effectively with their engineering partners.

#1: Maximize Value Rather Than Reduce Cost

Optimizing everything isn't always the answer: you may have a local optimization opportunity, but not one that globally helps the business. The engineering perspective is that finance is hyperfocused on spend. If finance is able to reframe themselves as being focused on business value, then engineers can relate more closely to that and will be more open to trade-off conversations aligning with the FinOps principle of decisions being driven by the business value of cloud. Engineers are comfortable making trade-offs—it's a core part of their job.

Gabe's take on this: "You want to enable them to maximize value. It's not just about spending less, it's about spending on the right things. For example, in a manufacturing example you might decide to use a more expensive material in order to get more out of it."

Aim to show engineers how they can add value to the company, and then ensure that they are recognized for doing so by enabling them to say, "We were able to get performance increases while also driving cost decreases." Anyone can throw more money at a performance challenge, but the really hard problem is to enable cost-efficient performance.

The Quality Equation

W. Edwards Deming, a renowned engineering professor of the mid-20th century, baked cost firmly into the denominator of the quality equation:

$$\text{Quality} = \frac{\text{Results of work efforts}}{\text{Total Costs}}$$

Quality engineering—and all its synonyms like cost engineering, financial engineering, etc.—is a means to an end: business value. Ultimately everyone is responsible for value, not just a single team or person. Deming's philosophy has been summarized on Wikipedia as follows:

> By adopting appropriate principles of management, organizations can increase quality and simultaneously reduce costs (by reducing waste, rework, staff attrition and litigation while increasing customer loyalty). The key is to practice continual improvement and think of manufacturing as a system, not as bits and pieces.

#2: Remember That We Are on the Same Team

Teams need to collaborate closely for FinOps to flourish. In his own organization, Gabe related the partnership to a bowling analogy: "Finance is the bumpers and engineering is the bowling ball. Together, we are aiming to knock down the most pins in the least amount of turns."

For many, the push to start or accelerate adoption of FinOps comes out of the finance side of the house, who push their engineering counterparts to get on top of costs and chase them with reports telling people to go reduce waste. Too often this comes with an accusatory tone and without enough understanding of how the cloud actually works to understand why engineers were making the choices they made.

#3: Prioritize Improving Communication

Gabe went on to say that "a lot of the terminology used in finance doesn't register with engineers. Likewise, engineering terminology doesn't register well with those used by finance. It's so important to be on the side of the engineer and not be against them. I look at FinOps work as a partnership where we also must help the nonengineers, like those in finance, understand what it looks like to partner with engineering. And to help both understand how the FinOps culture will meld into their daily lives: FinOps is not a separate thing, it should blend in until it's a habit just as security work must become embedded into the process." The importance of a common language around FinOps and cloud was covered in Chapter 4.

He proposed that organizations consider moving some engineering talent into finance, and alternatively have finance participate in some engineering activities so they can more directly understand the constraints engineering has to consider during trade-off conversations.

Stories from the Cloud—Benjamin Coles

Benjamin Coles, an Engineering Manager from Apple, offers this advice:

> Accountability seems like a simple term, right? You write code, you push code, and when the day is done, you clock out. Of course, it's never as easy as that. There are efforts to understand and write requirements, betas, bugs, regression, outages, and general performance issues that inevitably show up and must be dealt with—stat. And on top of all that, there's someone telling you how much money you're spending and that you're the one who's on the hook for fixing it.
>
> It takes lots of time to juggle all the work you're doing in parallel with the demands of others, and it just doesn't seem to get easier. But here's a secret: the people you're talking to in FinOps may have the levers you need to pull to make your life easier. They're trying to accomplish a goal—one that serves the company—and if you take a closer look, you'll see that it has the potential to benefit your own efforts as well.

Let's say, for example, that FinOps says doing something will result in large amounts of savings, but it will require significant effort. In a case like this, first confirm that the level of effort (LOE) does not exceed the potential savings, then explain the level of effort required and ask FinOps to negotiate dedicated time with your manager.

You can negotiate for extra people, extra projects, and efforts you think are important to you and your group. In many cases, the central FinOps team receives pushback, but they are often asking: "Can you help me help you?" You can counter-negotiate with resources or say, "Can I perform the top five recommendations versus the whole list of recommendations? Alternatively, can we defer this task two quarters from now to better fit your schedules?"

You might say these are all things for your manager to think about, but FinOps is a new concept, and FinOps teams need champions like you to help explain it to their managers. Sure, we know that we are inundated with a huge list of tasks, and the last thing you want to do is to add more items to that list. But understand that your manager is no different, and they're seeking your help to understand and assess how to move forward as a group. The payoffs are potentially great. Not only will you help your own organization, but you can take the skills you learn in managing costs to another company—helping you with your own career ambitions.

As an engineer, I suggest that you keep a running tally of costs you've saved the company. Finance and the business teams already do this, but you keep track of this data and you might find yourself quickly becoming a leader in this space as there aren't a lot of engineers fighting for the cause. Accountability works both ways, so hold others to the same level of standards that you hold yourself.

The key question finance partners should ask engineers is: "Do you have everything you need?" The key questions engineers should ask business or finance is: "Are the cost-saving changes going to be greater than the cost of the engineering effort it takes to make the changes? What other options can I explore to make this successful?"

Your job is to balance the cost versus the benefit, then make a decision. If someone asks you to clean up something small, it can be done as a five-minute effort of good-will. Or it may be simply that a call was made to deploy suboptimally on purpose and it will be ironed out in the next release. Whatever you decide, be accountable: always do everything in your power to follow up your commitments with action.

#4: Introduce Financial Constraints Early in the Product Development

If you introduce strong financial constraints at the end of the development cycle, do not expect them to be in the next release. Quick wins might move the needle in the right direction, but they are unlikely to get you to the long-term goals (see the Afterword of the book for more on prioritizing long-term gains over short-term wins). Financial constraints take time to implement, especially if introduced late in the game, as they may not be compatible with fundamental choices made early on, requiring rearchitecting. The cost to rearchitect may ultimately not be worth the cloud cost savings when compared to the engineering time required to do so and its impact on delivery schedules of other releases. As Gabe said in his FinOps X talk: "If finance only shows up at the point of a release hitting staging or production, then it's

too late. You need to bake cost constraints into the beginning, otherwise you're often asking engineering to build a new product."

#5: Enablement, Not Control

Requirements are the main thing engineers focus on. Their job is first to cut out the noise of the requirements to determine what really needs to be built. Then they examine the constraints to determine how to build it. If you have overly rigid restraints, you end up with what Gabe calls *technicians*: those who are told to simply "turn the valve this much" without ever innovating. On the flip side, you have what Gabe called *artists*, who go too far in crafting artisanal and usually overly expensive solutions to a problem. There's a classic episode of *The Simpsons* where Homer is given free rein and an unlimited budget to design the perfect car. What comes out the other end is an expensive monstrosity. Without finding an appropriate level of cost guardrails and other constraints, innovation can actually suffer.

In the middle there are enough constraints to be creative, but it's not so rigid that you can't find an elegant solution. Simple guardrails like a total cost or a list of approved cloud service SKUs or approved regions can create innovation. But if you do that, you need to provide other parameters where engineers have flexibility. For example, if you stay within this budget and avoid these services or regions, then you can do whatever you want.

On this principle, Gabe ended by sharing that "we are so worried about artists but we also don't want technicians. Too many constraints turn into restraints. Enable people to do the good behavior you want. And remember that all engineers do not know everything about the cloud; education is key."

#6: Leadership Support Isn't Helpful, It Is Essential

Leadership is the single most important factor of the success of a FinOps transformation, or any organizational transformation for that matter. If your engineering and finance leaders don't agree on the importance and the desired outcomes, then the cultural change is doomed to fail. Again, it's not that you need mandates from leadership but that you need leaders to constantly send signals that cost—and its ultimate outcome of business value—is important at all stages of development. The message should not be a mandate such as "reduce cost by 10%," but rather a constant drumbeat that "cost efficiency is always important and is a core part of your job."

Here Gabe referenced his former job as a security engineer, saying, "You can't just do occasional security sprints; you're always baking security into every part of the development lifecycle. If a new feature breaks your cost constraints then it's a bug, just as it would be with security. You have to apply that level of discipline."

Stories from the Cloud—Mike

My central FinOps team has evolved their approach over the years to work more closely with the hundreds of engineering partners we have at Atlassian.

We go to them now with a specific ask, such as: "We want to commit to some Savings Plans or Reserved Instances to achieve this specific goal. Here's the commitment that we are considering, here's the value that we expect it to return, here's what it will do for the business. Do you have any blockers to us moving forward?"

Over the years, more attention and time was given to us as we became better at focusing on value conversations and the FinOps team became more established. In the beginning, we were simply focused on best practices. I spent a lot of time trying to convince folks informally to think more about cost.

Once we solidified ourselves as a dedicated team with broad leadership support, it made it more obvious to the larger organization that FinOps is not just a side thought for Atlassian, as it was in the early days. They see that our team is being funded as a key investment by leadership, and this helps drive the message that they need to partner with us.

At the same time, we come in cautious about what we're trying to achieve. We make sure they understand the ask, what they need to do, and what the impact of that effort is.

The more challenging conversations happen when a team has made a budget commitment and now they aren't meeting it. In that case, we'll come in with questions like:

- Why are we not hitting the things we committed to?
- Is this a one-off?
- Is this a new trend?

We try to come up with a solution with them by asking questions like:

- Can you change some other things in order to hit the commitments?
- We see growing waste recommendations; can we slot some time to tidy them up?
- What can we change to reduce the likelihood of resources becoming waste?

When we need to be the police, we come in with information or data that proves there's something to be concerned about. Anomalous spend, an exceeded budget, or a forecast that shows that we expect to exceed the budget considerably. We ask questions like: this data indicates something is wrong, can you help us understand it? This slowly builds the rapport so that—instead of booking a meeting—we can eventually send a team a quick asynchronous question and get a cooperative response.

We still get pushback on certain requests, such as an ask to go clean up idle resources when there is not enough perceived time value. But these result in conversations, not arguments, because we have built trust and we have leadership support.

Data in the Path of the Engineer

Engineers—especially developers—often operate in cycles of deep focus, often referred to as *flow*. Interrupting one of these cycles can greatly impact the engineer's performance. It's for this reason that you should interrupt engineers only when the timing of the response or action needs to be immediate; using real-time communication methods such as chat, video call, or in-person walk up will pull engineers away from their train of thought and disrupt their focus. If the action you need them to make can wait, then consider how you can inform them of the task in a way that won't be disrupting.

Returning to the concept of *data in the path of the engineer* introduced in Chapter 8, having FinOps data exist in the places your engineering teams are already working is a key contributor to reducing the effort needed by engineers to engage with FinOps. Integrating with engineers' existing processes avoids the effort required for them to open a new set of reports and run additional rituals around FinOps. However, injecting FinOps into engineering team reports and processes unannounced should be avoided, as this can really interrupt the flow of engineering team rituals. Ideally, engineers should help guide where inserting FinOps data will most help them within their existing work processes.

If leadership mandates to their engineers that they must clean up cloud waste immediately and cut costs by 10%, it's likely they will see quick results. Mandates are a quick way to get action from engineers. But they are often not sustainable and can create a negative culture of whiplash. As soon as the mandate goes stale and is no longer being reinforced by leadership, it falls out of the minds of engineering teams, who fall back into focusing on delivery and optimization of other key metrics. Whereas the negative feelings toward FinOps are long-lasting.

Cost mandates are point-in-time impacts that can be effective in the short term, but you will run into the same creeping waste again over time. The broader message that cost metrics are important to the business should be continuously reiterated as part of the way engineering operates.

When you start to build in at the cultural level that cost efficiency is a core value, it becomes part of the workflow of services. Aim for rehearsed practice and building muscle memory. FinOps becomes part of the whole story. Culture doesn't go away—it just becomes part of the day-to-day.

Partner with engineering teams to understand their existing processes and reports, gaining feedback from them on how and where FinOps data will be of the biggest benefit. When engineers are able to contribute to how FinOps is implemented, they are more likely to help make the needed changes.

Models for Partnering with Engineering Teams

Each FinOps team is going to partner with their company's engineers in different ways, driven by the mix of knowledge and skills of the members on the FinOps team.

Direct Contribution

FinOps teams with engineering skills often take a more proactive approach by directly helping engineers design cloud services to be more efficient and work on tools that avoid creating waste up front. This is a powerful method of building a collaborative environment between engineers and FinOps teams that can be more effective at keeping the amount of usage optimization to a minimum. For very large companies, this type of partnership can have its limitations, as the number of engineering teams actively working on cloud can overwhelm the capacity of the FinOps team.

Indirect Collaboration

Not all FinOps teams are staffed with engineers, but this does not mean that they will be ineffective at partnering with the engineering teams around them. This type of partnership will often see strong collaboration between the FinOps team and the Cloud Center of Excellence (CCoE) or Technical Architecture Group (TAG) within the organization. Lean on the TAG and/or CCoE teams to help drive all engineering teams to design more efficiently and use their deep cloud knowledge to help FinOps understand current cloud spend, predict future costs, and clearly identify opportunities for optimization.

Indirect Collaboration with Targeted Contribution

As a hybrid approach to the partnership with engineering teams, take the indirect collaboration model but use the engineering skills of the members of the FinOps team to target specific areas of the business. The FinOps members could work directly with your engineering teams with the largest spend or focus on the least cloud mature engineering teams within your organization to uplift them. The FinOps team members performing this direct collaboration may float around the organization as the need changes or be dedicated to a specific engineering domain. At Atlassian, there are *cost engineers* who embed within other teams to perform this function.

Picking a model to build your partnership should be done by first assessing what skills are in the FinOps team, how well established your CCoE/TAG teams are within engineering, and the maturity of cloud adoption across your organization by all engineering teams. Choosing one model does not lock you into that way of partnering forever, and as your cloud footprint scales, the hybrid model becomes more and more likely to be where you end up.

Conclusion

It is our hope that you've gotten a better understanding of the engineering persona and the central role it plays in FinOps. Partners from all sides need to listen and break down challenges that aren't necessarily engineering-specific. We can't lob a challenge over the fence and expect it to be pushed to completion. Engineers have a huge set of competing priorities they are juggling. When cost awareness is introduced, additional cognitive load comes with it. The method and timing of communication with engineers can greatly impact the perception of FinOps being a partnership versus a mandate.

Friction around making cost-efficiency improvements typically comes from one of these areas:

- Other priorities being set by their leadership
- Tight deadlines to deliver features or services
- Concerns that changes might have negative consequences for performance
- Unclear expectations for engineers, or the value of the effort is not well communicated

To summarize:

- Building a partnership with engineers is more important than getting a list of FinOps tasks completed.
- Cost savings is not the goal of the partnership, but rather efficient value for money spent.
- Align with engineering teams on the financial constraints early in the development process to reduce wasted effort later.
- Leadership should aim to avoid mandates (i.e., short-term cost focus periods) but instead encourage teams to continuously monitor and manage the needs of FinOps as part of the workflow and culture in the organization.

- If your FinOps team doesn't have the skills to review/recommend engineering designs/changes, collaborate with the architecture team(s) within your organization.

The next chapter looks at some other types of partnership in the organization, interacting with teams managing other IT frameworks.

CHAPTER 25
Connectivity to Other Frameworks

Unless your company was born in the cloud and spends most of its technology budget there, FinOps likely won't exist in a vacuum, and it will probably not be the only IT management discipline your organization practices.

Most large organizations use a variety of different IT management disciplines to define how they will deliver technology services, to understand the value of the spending associated with them, to accelerate the adoption or delivery curve, and to keep the organization safe from a wide variety of risks.

This chapter lays out a starting path for how to work with the teams in your organization using other IT and technology spending-related frameworks. The focus will be less about the technical intersections between the methodologies (that's more detail than we have room for here) and more about the ways to work with teams/people who do them. The goal is to work with these other frameworks and disciplines successfully, to complement them instead of overlapping or competing with them. Too often we've seen friction between disciplines or offices emerge as each tries to establish power over the other or create influence within the organization. FinOps is not meant to be a panacea to all IT spending problems, but to help enable all disciplines in the organization—including other IT management functions and teams—to manage cloud use and cost effectively.

The exhaustive list of IT-related frameworks is long and—for many of us—confusing. They cover a wide range of IT management areas, such as assets, licenses, financial management, security, development practices, architectural processes, and more.

Here are some of the common frameworks we see alongside FinOps. They typically interact with FinOps in the following ways:

IT Management disciplines
: FinOps can share details like what things are running within cloud environments, how cloud resources are configured, and in which locations/regions they are running. Items covered in Chapter 12, including cost allocation, account hierarchies, and tags, are important here.

  - IT Service Management (ITSM)
  - IT Infrastructure Library (ITIL)
  - Configuration Management (CM)

IT Financial disciplines
: FinOps can help classify the different types and classes of cloud usage seen within the cloud bill. The needed classifications may also drive your tagging approach. Tracking license usage across cloud environments and the lifecycle of assets are important concepts. They focus on matching the deep data of cloud billing with the existing categories of IT costs provided by other finance frameworks.

  - IT Financial Management (ITFM)
  - Technology Business Management (TBM)
  - Software Asset Management (SAM)
  - IT Asset Management (ITAM)

Information Security / Cybersecurity
: FinOps often shares anomaly detection data that may help alert security teams to areas of concern. Tagging standards can help security teams identify the ownership and intended purpose of cloud resources quickly when flagged for review by security processes.

GreenOps
: This is covered in detail in Chapter 19.

These methodologies, disciplines, and practices may have been functioning within your organization for many years to help govern specific aspects of IT delivery prior to your use of public cloud. A common *a-ha* moment is the idea that these other disciplines are not actually equipped to deal with the self-service, on-demand, and massively variable nature of cloud, nor with the sheer volume of billing data it puts out. FinOps isn't meant to replace these but rather to help deliver the same general outcome: the maximum value for every technology dollar spent, but with a deep domain focus on cloud.

Your organization may be adopting or using one of the many implementations of Agile or DevOps frameworks to help accelerate and coordinate your IT value delivery. Your organization may have enterprise architecture teams leveraging one of the many architecture frameworks to drive how you develop and implement your

IT solutions. You may already have a Cloud Center of Excellence (CCoE) or cloud platform team using one of the Cloud Adoption Frameworks or Well-Architected Frameworks from the cloud service providers you use. And you certainly have an information security team focused on ensuring that every product and engineering team maintains a keen focus on cybersecurity requirements and risk.

Security is a phenomenal parallel to the FinOps team: the security team is not solely responsible for ensuring secure practices around the organization but rather for providing guidance, centralized best practices, and acceleration to the line of business teams.

Not every company will have all of these other groups. In established organizations, the FinOps team may well be the newest of many groups to arrive on the scene (save for GreenOps, discussed in Chapter 19) to help change the culture of the organization and sharpen the value that IT can bring the organization as a whole.

Total Cost of Ownership

Because FinOps is particularly focused on the cloud, you'll need to engage other teams in the organization to get the total cost of ownership (TCO) for an application or portfolio that exists partly in cloud and partly on premises. All those other costs that you don't buy through the cloud bill—rent, labor, contracts, and some licenses—and as a result don't typically end up in your FinOps reporting, are all part of your TCO. Understanding TCO is critical to determining the value from your technology purchases and enabling the nirvana state of data-driven decision making that we discuss in Chapter 26.

There are peculiarities to the cloud that make FinOps necessary—the huge volume of data you're dealing with, the speed with which it's coming to you, and the unlimited availability that creates the opportunities for overspending—and that FinOps is uniquely equipped to handle. If your finance team is already managing those *monthly* or fixed costs of IT, the FinOps team can blend in the cloud costs in a usable format to give the organization a full picture of spend by application, by business unit, by product, etc. The FinOps team can go deeper to provide not only the allocation of that spend but the story of why it is what it is, and unlock potential optimizations that require finer-grained visibility than many traditional IT management disciplines can do, which they need to continue to make IT spending more effective overall.

Working with Other Methodologies and Frameworks

There are a number of steps you should take as you develop FinOps to avoid conflict with other methodologies or frameworks that are already established.

Over time, as the cloud becomes the majority of IT spend for most companies, some of these existing frameworks may be consolidated to operate as one. Recognize this in the beginning and be on the lookout for opportunities to combine efforts so that your organization gets the most benefit in the long run. Focus on the outcomes that each framework is designed to achieve.

Find Out Who's Out There

It is not uncommon for a FinOps team to begin their work and then sometime later discover that there are other teams operating other IT frameworks. This can lead to a perception between the teams that each is conflicting with what they are trying to achieve or duplicating efforts. Ideally, you identify who is running which frameworks and functions early on in the process of implementing FinOps. Seek first to understand what each team does and what value they are delivering for the business. Go talk to them, find out what they are trying to accomplish and who is their executive sponsor, and, above all else, begin to build a working relationship.

Stories from the Cloud—Ashley Hromatko

During her time as FinOps leader at Pearson, Ashley Hromatko shared the following story on how her team engaged and built a cooperative practice with an existing SAM practice:

> At the height of our digital transformation journey and migrating hundreds of workloads to cloud, we began to shift and simplify our procurement of third-party software and services to the AWS Marketplace. Our organization had centralized Procurement and SAM teams, but the businesses wanted to unlock innovation with faster procurement in the cloud. This agility came at a risk that the businesses were accessing funds without approvals or negotiations, buying licensing without insights into company-wide deals, and concerns around proper vetting of vendor contracts.
>
> Our team—personifying the FinOps principle that "Teams need to collaborate"—set out to learn the existing Procurement and SAM processes and began advocating on behalf of the engineering business units about the benefits of efficiency, commitment advantages, and granular tracking that could be accessed.
>
> Together these teams were able to establish some rules of the road and new cloud third-party governance was jointly established. This included locking down access to AWS Marketplace to authorized leads, creating onboarding guidelines with thresholds that required SAM and finance engagement before purchase, documenting a qualified vendor list, setting up workflows that collected details prior to purchase, regularly auditing purchases, and a new amortized cost allocation output. We continued partnering together annually on forecasting and making business decisions for the next year on where to procure what type of third-party software and budget thresholds.

Make Friends and Share Goals

Create a shared understanding that there is not a competition to see who does the best job. Having an *us* versus *them* attitude (discussed in relation to engineering teams in Chapter 24) will only lead to both sides being less willing to collaborate, in the same way it does when FinOps and other teams like engineering or finance see themselves as having opposing goals. The perception is that the success of one can often come at the expense of the other. However, the shared goal should be the success of the organization's mission.

Overlapping goals between the teams are expected, so clearly defining those will enable a shared roadmap to avoid duplication or, even worse, conflict between them.

Some areas to explore for overlapping goals include the following:

- Who is supplying the data needed to make timely decisions and how that data allows the business to measure trade-offs designed to maximize business value
- Which team is responsible for each portion of the IT spend and usage management
- How IT spend is to be controlled/governed and who defines it
- How the data, reports, and dashboards each methodology provides will be generated and distributed

FinOps doesn't do it all—and it's not intended to—nor does any other framework. Identifying each framework's strengths and weaknesses will enable you to identify where you can help each other with the larger organizational goal, with the outputs of one framework solving weaknesses for the other. Drawing up a responsibility chart can help the entire organization understand how each team works together.

Here are some areas where FinOps is often different than other frameworks:

Scope
: Other frameworks are often aimed at all IT-related spending, such as staff costs, consulting contracts, license management, and data center operations, whereas FinOps has a specific focus on public cloud and SaaS providers with variable spend costs specifically.

Scale of inputs
: Other frameworks often work with a large number of source inputs, each of lower scale but varying in complexity and consistency. FinOps relies upon a limited number of source inputs containing extremely large and complex datasets.

Speed

Other frameworks work best with traditional cost elements that operate on longer business cycles (monthly, quarterly). FinOps is designed to be executed in an action cycle as quickly as possible (hourly, daily) to react to the variability of cloud usage.

A solid combination of frameworks can create a *T-shaped* approach to technology value delivery. In other words, together they deliver broad visibility via frameworks like ITFM with a wide (but less granular) scope as well as granular domain abilities via frameworks like FinOps with a deep (but less wide) scope.

Share Influence, Terminology, and Processes

Existing framework teams are likely to already have influence over the same personas you will need to influence with the introduction of FinOps, or the growing interest from your organization to introduce FinOps may offer an opportunity for existing framework teams to amplify their impact, as in Ashley's story about interacting with the SAM team. Teams that have existing relationships across the organization can help introduce FinOps and shorten the process of FinOps becoming a known practice within the organization. Aligning on the common messaging across the frameworks means you end up with a shared voice.

All of you are doing things that:

- *Are not* the core business of the organization, i.e., the time and effort spent implementing these frameworks is time and effort not spent building the products that your organization is selling
- *Require work* and collaboration on the part of many people throughout the company
- *Are critical* to the success of the organization, i.e., without these frameworks the organization will have challenges that will negatively impact its success

Aim to build a shared influence across your company that both expresses the importance of each relevant framework and helps support why both are required for the organization's success.

To implement any of these frameworks, there are datasets, reports, and education of domain-specific terminology required. Before implementing a new policy or standard upon your organization, you should identify existing policies and standards that can be reused or repurposed. Need a tagging plan? An existing framework team like ITFM, TBM, ITAM, DevOps, or your CCoE may already have some allocation rules defined. Leaning on existing processes and knowledge in the organization will make things simpler and speed up the process of building a mature practice. Perhaps more important, adopting terminology and definitions that are well known inside your

company is much easier than trying to create new—and potentially conflicting—uses of similar terms.

Share Infrastructure

Remember the concept of *putting data in the path of each persona*, covered in Chapter 8. Existing frameworks are likely to have existing reports and dashboards that engineers, finance, and leadership are already using. Instead of making new reports and dashboards, you should align with the teams who own these existing reports and add the FinOps messaging. If ITFM reports are already a part of your executive reporting processes, aim to add deeper cloud data via your FinOps practice to those reports. Don't try to reinvent the wheel. Sharing existing reports and dashboards both reduces effort in establishing FinOps reporting and avoids extra effort being required by the relevant personas across your organization.

Share Knowledge

FinOps is an integrative discipline that ties together the work of many other disciplines in your organizations, so there are several key disciplines that have key similarities to FinOps. Oftentimes practitioners of these disciplines describe FinOps as "my discipline, but for cloud." But there are important differences in cloud, which is why FinOps needs its own focus in most organizations.

Without understanding the strengths and weaknesses of existing frameworks that we discussed in sharing goals, it's easy to trivialize what each is and what it can achieve for your organization. For FinOps, it is a mistake to think that existing frameworks have nothing to offer. Plenty of the activities they already perform will be delivering value that can be leveraged by FinOps. Sharing education programs, outreach, and evangelism builds a common understanding of each other's frameworks; this will enable all teams to have a shared appreciation for the others' frameworks.

Conclusion

FinOps is about building bridges, not walls. All IT management methodologies have a common goal of helping the business get the most value out of IT spending, but they approach it from different angles, and some are specialized to target certain types of IT spending.

To summarize:

- Seek to identify teams inside your organization that are operating other frameworks.

- Aim to align and complement teams in your organization managing other frameworks. You all have something to learn from each other and will find areas where you can support each other.
- Work together to both avoid duplication and empower each other's goals.
- Decide which team will be responsible for each component of overlapping responsibility.
- Build consistency in your messaging inside your organization around cloud spend and business value decisions.

Now you have an understanding of how to build relationships and truly collaborate with engineers, finance, and the many other groups your FinOps team will work with. Now it's time to make those relationships pay dividends.

This brings you to what we like to call *FinOps nirvana*. The penultimate chapter will encapsulate everything we've discussed so far that enables you to achieve the ultimate goal: enabling data-driven decision making.

CHAPTER 26

FinOps Nirvana: Data-Driven Decision Making

The nirvana stage of FinOps is when accountability for spending is fully decentralized to engineering teams who make efficient choices every day—not just when there's a mandate—empowered by leadership support who have made cost a first-class citizen alongside other critical software performance metrics.

The goal is getting to a place where your organization makes ongoing, data-driven decisions on cloud spend based on business value. Who makes these decisions?

- *Architects*, when they are designing infrastructure
- *Engineers*, when they are writing code and deploying services
- *Finance and procurement teams*, when they are making commitments to cloud providers
- *Leadership*, when they are driving technology strategy

It's an ongoing collaborative effort that can fully happen only when a FinOps cultural transformation has occurred.

Everything you've done to date has enabled you to get here. You've allocated spend, set optimization goals, and implemented procedures to enable your teams to reach those goals. You're continuously exercising the FinOps lifecycle, refining your allocation strategies, setting metric thresholds, and providing increasingly refined visibility into cloud usage each time.

But there's still a gap. Each time your bill goes up, the debate reopens about whether the spend is good or bad, i.e., whether the spend levels are the right ones for the business outcomes you're trying to achieve. Has the bill increased due to growth in

your business? Is it because of the acceleration of cloud migrations? Or has it gone up due to inefficient patterns of usage creeping back into your teams' habits? It's very tricky to give a definitive answer, especially one in which executives will have confidence.

In this chapter, you'll start tying your spend to business value, primarily through unit metrics applied to groupings of your cloud costs. These metrics help you to identify the value in allowing a certain amount of spend to capture additional business value, to set goals on cost performance, and, maybe most importantly, to be able to prove to your executives that the decisions your engineering teams are making are driving business value.

Unit Economics and Metrics

The first edition of the book concluded that unit economics was the nirvana state of FinOps. However, a few years ago, FinOps was mostly being done by larger companies not spending very much of their total IT spend on cloud or within smaller companies where their singular cloud focus and limited product suite made it easier for a single unit economic to drive decision making for the company.

As of 2022, nearly all enterprises use the cloud, many are using material amounts that affect their bottom line, and as a result FinOps has become a critical part of making important decisions that affect nearly every aspect of the organization. Watching this evolution, we can see now that unit economic metric data is just an input—albeit an important one—on the path to ongoing value-driven decision making by engineering and business teams.

To enable engineers to make these daily decisions about cloud spend, they need to have the right metrics presented to them. In the same way that you can't optimize performance without relevant metrics that illuminate user experience, you can't optimize cost efficiency without metrics that reveal the outputs you're getting from that spend. This is where unit metrics—and the broader concept of unit economics—comes into play.

A simple definition for *unit economic metrics* is direct revenues or costs, associated with a particular business model, that are specifically expressed on a per-unit basis. For a customer-facing application, that unit might be a user or customer subscription; for an ecommerce platform, it might be a transaction; and for an airline, it might be seat miles.

Based on the work by a FinOps Foundation unit economics working group of peers, a longer definition for unit economics is developing:

Cloud Unit Economics is a system of profit maximization based on objective measurements of how well your organization is performing against not only its FinOps goals but as a business in the market. Cloud Unit Economics achieves these noble goals by leveraging the measurement of marginal cost (a.k.a., unit cost metrics) specific to the development and delivery of cloud-based software and marginal revenue (a.k.a., unit revenue metrics).

There is not *a single metric* that the whole organization should follow. You must aim to do unit economics at multiple levels across the organization, with many different inputs and granularities. You can do high-level unit economics with metrics like cloud cost per customer, but you can also do unit economics at an application level by doing things like cost per transaction, or even do unit economics on an individual service level like compute cost per execution of a microservice.

Unit Economics Don't Have to Be About Revenue

We heard after the first edition of this book that some people thought that there needed to be a universal unit economic metric, a single north star metric for your organization to follow. This was not the intention then, nor is it the reality now. There isn't a single north star metric, but rather a constellation of stars that together form a map allowing you to safely navigate the turbulent seas of your cloud adoption journey in pursuit of calmer shores.

Don't be discouraged if you're unable to get to a top-line revenue metric related to cloud spend. Many organizations cannot, either because of the type of business they are in or because of their maturity. Unit economic metrics should function at multiple levels like a nesting doll. If you're a portfolio manager, you might have operational activity–based unit economics for each of your applications, or you might have them for a whole business unit, or as the lead on an application you might have a set of measures that tell you how much it costs for your application to take an action that isn't possible to tie back to revenue. Each may be very different and none may tell the complete picture of cloud value, but together they provide direction and can help you (and your executives) begin to trend your progress toward maximizing the value you are getting from cloud.

The important thing that unit economic metrics do is unlock the ability to talk about your cost in comparison to some value or business goal. They don't need to be a panacea that aligns each dollar of spend to each dollar of revenue. They can happen at any level and with nearly any operational business metric, however specific or minute. They enable value maximization decisions based on objective measures of how well your organization's cloud usage is performing.

Calculating Unit Economic Metrics

In every cloud unit economics calculation, there is a numerator and denominator. You're taking something (generally an activity or output metric) and dividing it by something else (generally a portion of cloud spend) to come up with a unit metric value.

The numerator is a cloud measurement of spend that quantifies the choices made by architecture and engineering teams. The denominator is either an engineering value measurement or financial value measurement or business value measurement that gives that spend important context. The outcome of that calculation is the cloud unit economics value measurement that you're going to use to manage some part of your business, or to give insight into what changes you could consider making.

Somewhere in the organization, some form of unit economics is already taking place. It may not be in your team or business unit. That's one of the first things that Anthony "TJ" Johnson, Cloud Business Manager from Box, Inc., recommends: to search for allies who are managing to unit economic metrics already. Typically it's someone in finance who may already be providing related data to a business unit owner measuring how well they're utilizing a technology investment. That becomes a potential measurement for you to include cloud information infrastructure, especially if you are a cloud-first or a cloud native organization and you're focused on that as a path for your future. Find the unit metric dashboard that the team is using, find out who makes that dashboard, and go talk to them about incorporating cloud spend data.

Finding the correct unit to measure your business is a fine art, and it's something you'll likely change or update over time. Organizations with complex structures and many product offerings will often have many different unit economic metrics, each one relevant to a specific product, application, or service.

On Episode 11 of FinOpsPod, Anthony "TJ" Johnson, Cloud Business Manager from Box, Inc., shared how he looks at maturity as it relates to unit economics. Listen to his approach to maturing unit cost integration and budgeting, ultimately focusing on a cost-to-serve approach.

Spending Is Fine, Wasting Is Not

Cast your mind back to the inform phase of the FinOps lifecycle. This phase focuses on surfacing cloud costs to build awareness and accountability for spending. Through this visibility into cloud spend, you're able to determine trends and build forecasts for future costs. When cloud costs exceed your planned budgets, the next obvious step is to begin investigating the increase and explain what is happening. In the same way

that other metrics give context to cost optimizations within the metric-driven cost optimizations of Chapter 22, a unit metric provides the critical business context to changes in cloud spend.

We sometimes see SaaS products start by using business revenue generated from products—run within the cloud—as a simple starting point. By dividing total cloud costs by the revenue generated, you're able to determine if growth in cloud spending is increasing profits for the organization—and is therefore *good* spend.

By calculating cloud spend for total revenue, you can attach growth in cloud spending to your overall business growth. When these are in line, it makes sense that cloud spend isn't wasted. When cloud spend is growing faster than the business, there may be cause for concern.

Ideally, the metric you use for unit economics should have low volatility, where decisions in one part of the business do not affect the metrics used by others. Let's take a look at an example.

In Figure 26-1, your company is tracking total organization revenue over cloud spend. As long as these lines more or less remain consistent, your unit economics appear to be in order.

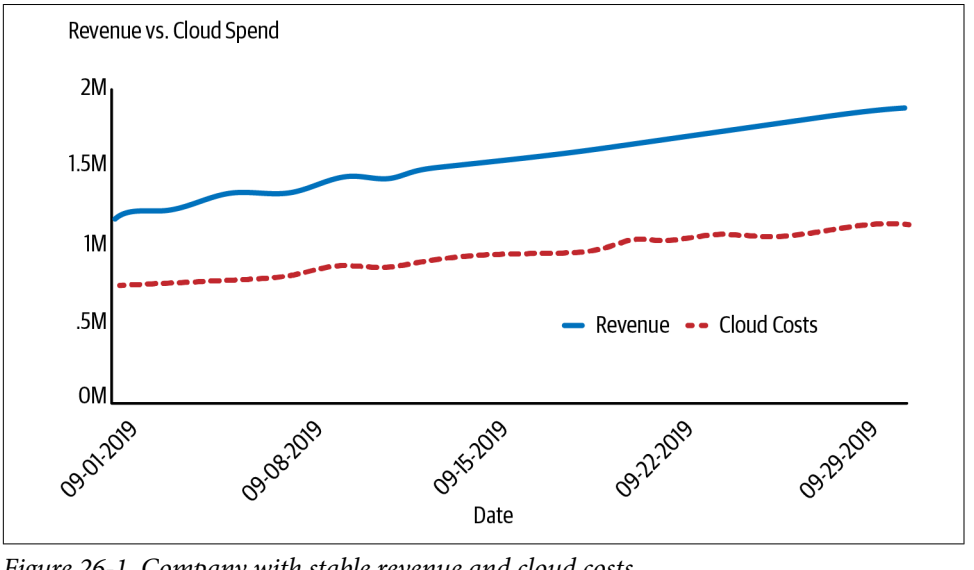


Figure 26-1. Company with stable revenue and cloud costs

In Figure 26-2, your business introduces a free tier to your cloud offering, allowing customers to sign up and use your service for free. The expectation is these free customers will eventually sign up for your paid offering and therefore increase overall revenue. However, when you introduce this new free offering, your cloud spend increases with no direct impact to revenue. Your unit economics have been affected.

If your engineering teams are using this metric to gauge how well their infrastructure costs align with the business value, this has a negative effect. Of course, you could just tell your engineering teams to expect this change and continue on—that is, until six months from now when your marketing team runs an advertising operation to drive up adoption for your free tier and your unit metric is affected.

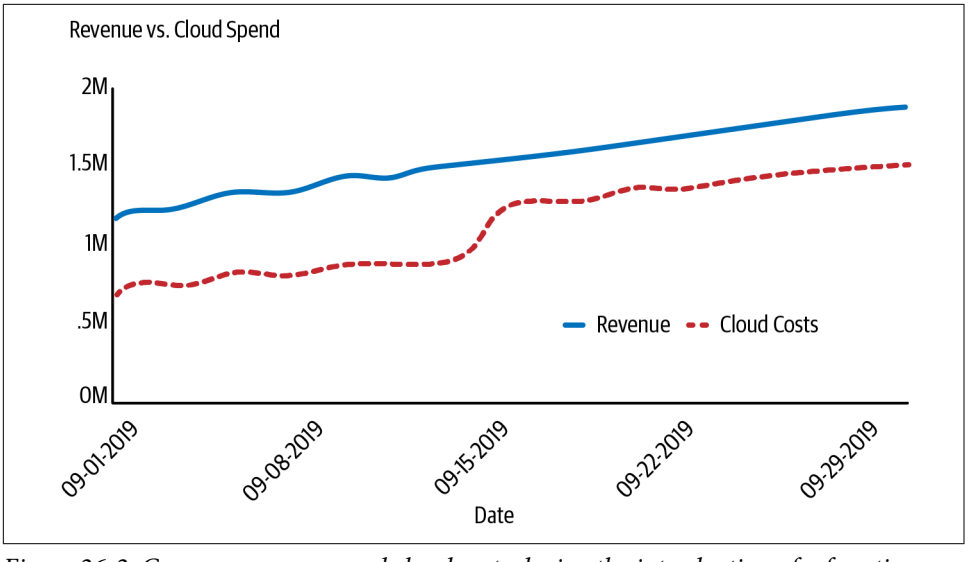


Figure 26-2. Company revenue and cloud costs during the introduction of a free-tier offering

In Figure 26-3, you decide to measure cloud spend against *monthly active users* (MAU). As you increase your active users, your cloud spend is likely to increase as well. With the introduction of the free tier, both active users and cloud spend increase together and your metric remains consistent overall. In this case, if a future marketing campaign drives more customers, it would have a similar effect to the graph and remain consistent for engineering teams. They can see that an increase in active users increased cloud spend. In this metric, the business value (active users) is measured against the cloud spend.

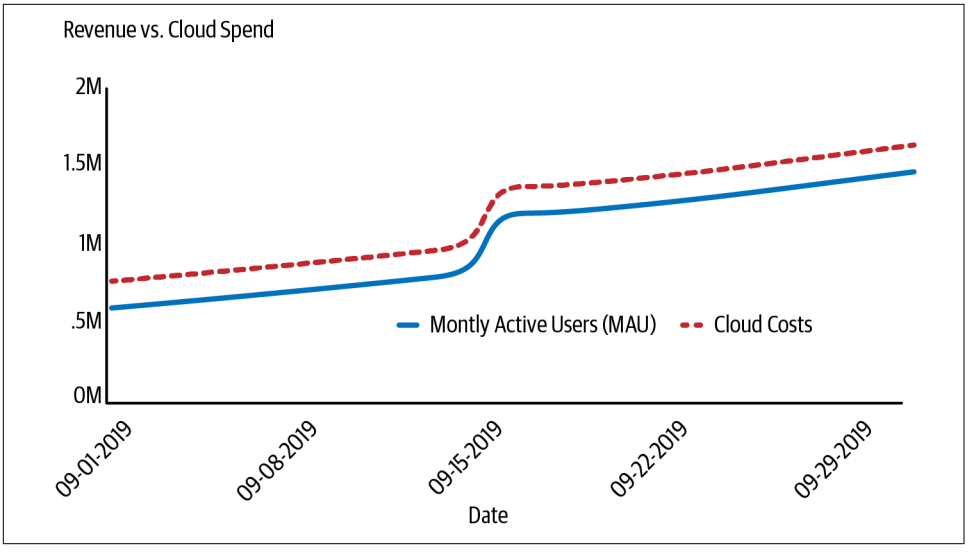


Figure 26-3. Monthly active users and company cloud costs during the introduction of a free-tier offering

Changing from corporate revenue to a unit like MAU, pages served, or API calls made will provide better context. Changes in the price charged to customers won't change the unit metric value when you're using a unit that's not directly tied to the revenue of the organization. When you use these metrics, measuring the amount of business throughput created by the cloud platform—especially in relation to cloud spend—becomes the key element in making business decisions around cloud cost efficiency.

Both of these unit economic metrics are useful for different purposes. Figure 26-2 indicates the organization has some work to do in effecting conversion from free tier to paid—not something the engineering team is responsible for—while Figure 26-3 gives us information about how well the engineering team is controlling costs. These are both important for the business but highlight different decisions that need to be made by various parties influencing and relying on cloud spend.

Activity-Based Costing

You're probably thinking: "Well, that sounds fine, but my business isn't so simple that I can align cloud spend to revenue." What type of unit metrics should you provide if your business model or structure makes tying revenue to cloud challenging or impossible? This is where another form of unit metrics called *activity-based costing* aims to look at the units of nonmonetary outputs that a set of cloud spend is creating.

Look at what tasks each specific application or service is performing. Is the application servicing API calls, is it returning a certain unit of data, or is it moving something from one place to another? These measures are often even more effective than revenue in enabling data-driven decisions by the long tail of teams and applications in the cloud.

Advanced practitioners will look at what it costs to perform specific tasks related to the underlying functions of the infrastructure. They budget with an understanding of the machine hours involved to accomplish the task: the cost per file created, cost per render, etc.

In Figure 26-4, you're tracking the number of files being rendered by your service and the cloud costs involved in supporting it. As render jobs increase, so do the cloud costs.

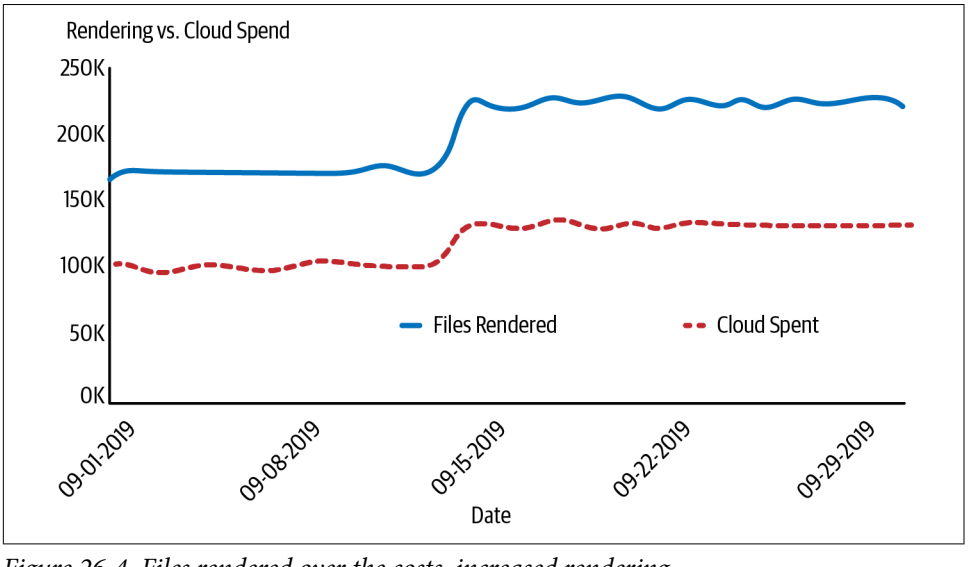


Figure 26-4. Files rendered over the costs, increased rendering

In Figure 26-5, you're tracking the costs of rendering files. The cloud costs increase mid-month, but the number of rendered jobs does not. This highlights that something has changed. You're now paying more to render a file than you were at the start of the month. This could be due to more resources being used or reservations no longer being applied to your cloud resources.

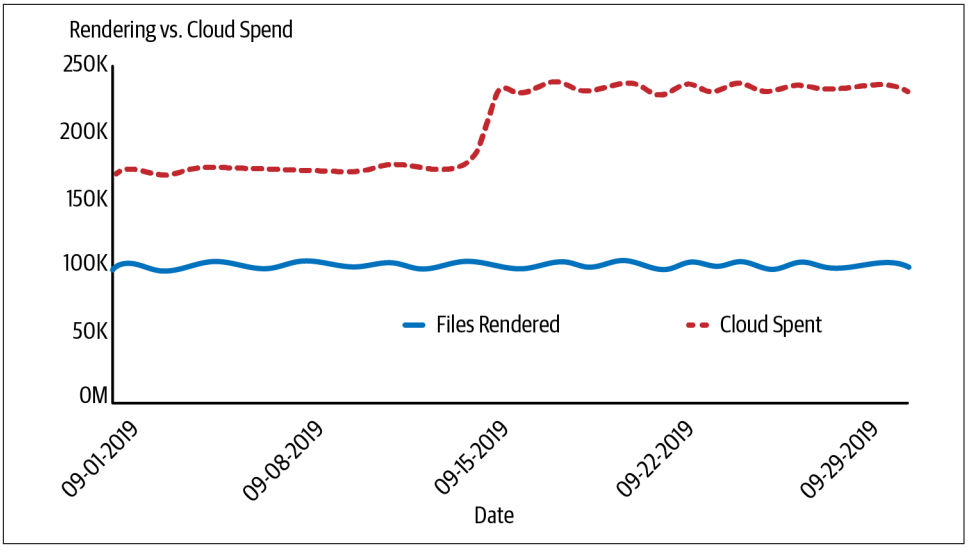


Figure 26-5. Files rendered over the costs, increased cost

As we discussed in Chapter 5, the cloud bill is driven by the confluence of the *usage* multiplied by the *rate*. So when you're over budget, you can determine whether it's about your resource usage or changes to your commitment-based discounts. When you're doing activity-based costing, you're actively managing those cost drivers.

Coming Back to the Iron Triangle

One of the best ways to enable your organization to make decisions is by using the Iron Triangle (i.e., balancing between good, fast, and cheap) discussed in Chapter 14 to frame these decisions.

Unit metrics provide teams the data they need to measure how infrastructure changes will impact costs. Conversations between technical operations teams and finance become less about the technical details of the change and more about the impact they will have, freeing up engineers to focus on working within the unit metric-enabled financial constraints agreed with the business teams, as discussed in Chapter 26. This allows the business to make decisions that weigh the benefits of the change against the expected impact to the unit metrics, such as whether increasing the cost per active user by 2% might boost application performance by 10%, which could have positive business outcomes that make the additional investment worthwhile.

We hear of some FinOps practitioners choosing not to implement unit metrics or delaying them due to their complexity. Ultimately, the nirvana state of FinOps is for your organization to be making decisions on the value of the cloud spend, no matter what method you use to measure value and compare it to costs: deciding when to

spend more in order to have a positive impact on the value you get from your cloud environment, and when to reduce cost is more appropriate.

When you apply unit metrics to the optimize phase of the FinOps lifecycle, your goal setting matures from using only operational metrics, like hitting 90% commitment-based discount coverage, to using revenue-impacting goals like reducing cost per subscriber by 25%. The former could help the business, but it can also be a distraction from what's really important.

What's Missing from the Equation?

So far, you've been using cloud spend as the numerator; however, you will likely need to incorporate a number of other costs into the TCO. After all, infrastructure costs often pale in comparison to labor costs.[1] Unit metrics are most helpful when you have all the relevant cost data, including things like labor costs.

The successful FinOps stories we've shared so far focus purely on cloud costs. Dividing the cloud spend alone by a unit provides limited context to the real cost per unit. In reality, a business has many other costs contributing to the total cost of generating revenue.

In Chapter 15, we discussed refactoring services into a serverless architectural model. When deciding to rearchitect services, you should weigh the infrastructure charges and potential savings against the labor costs for refactoring. Also, once a service is running in a serverless architecture, the cost of the service can become less about the cost of the cloud resources being charged and more about the operational costs of labor.

Of course, you could extend FinOps itself to track costs from outside the cloud bill, like costs of the equipment still running in your data center or the added costs of software licensing. However, in our opinion, it makes better sense for FinOps to work alongside existing financial models as we discussed in Chapter 25. Working with these other frameworks, you can enrich the unit metrics you use to make decisions with the total cost, instead of one that includes only cloud spend.

Using the FinOps maturity model as a guide, here are typical phases of bringing in noncloud costs to the unit economics denominator as laid out in the *Introduction to Cloud Unit Economics* paper from a recent FinOps Foundation working group:

1 Matt Asay, "Labor Costs Can Make Up 50% of Public Cloud Migration, Is It Worth It?" *TechRepublic*, May 9, 2016, *https://oreil.ly/76m57*.

Step 1: Cloud Costs Only
: Depending on the complexity of the application, this may also be done in multiple phases, where some cloud costs are handled initially as "direct" and others "shared."

Step 2: Cloud + SaaS + Licensing Costs
: This could include things like Datadog, ServiceNow, or PagerDuty, or BYOL ("bring your own license") like Windows or SQL Server, which are run on cloud infrastructure. Once you start bringing in SaaS tools or licenses, you may need to work with the team responsible for that product to understand the KPIs they care about and how they quantify time spent, such as the SAM or ITAM team.

Step 3: Cloud + SaaS + Human Capital + Hybrid Costs
: This is bringing in costs like labor or even on-premises costs that support a particular product or service. The run phase is typically when a firm starts collecting more than one metric for various parts of a complex system getting more granular over time.

When you begin including noncloud costs in your FinOps unit economics, such as labor and noncloud technology investments, you're not managing simply cloud costs anymore; you're managing the larger business.

When the FinOps culture allows all your groups across the organization to collaboratively look at all of your cost data, compared to the value that your organization is providing or deriving from it, and make good decisions, that's the nirvana state of unit economic data-driven cloud value decision making. It's not something FinOps can do alone. Cloud is increasingly becoming the driver of digital value in every organization of the future, so it is something that cannot be done without FinOps.

When Have You Won at FinOps?

We want to highlight that there is no finish line when it comes to FinOps. We can't offer you the point beyond which you have no more to learn and nothing more to do. Practices continue to evolve, and cloud service providers constantly release updates to their offerings and related billing data. In short, as long as cloud is evolving and changing, so will FinOps.

Companies implementing FinOps today—especially those early in their cloud journey—will derive the most value from their cloud investments by avoiding the bill shock moments and related innovation slowdown. FinOps in these organizations won't be seen as a single role or small *team that does FinOps*; instead, all related personas inside the organization (from engineering to finance to procurement and business) will understand their role to play in the *whole company doing FinOps*.

When the FinOps culture is working well:

- The centralized FinOps team is properly funded to solve the centralized functional activities of FinOps, reducing duplicated effort by individual teams and guiding the company along its ever increasing FinOps maturity.
- All teams communicate effectively with one another on a regular cadence, using automation and common platforms and a common vocabulary.
- The constant updates to cloud services, new offerings, and regular creation and deletion of systems according to value derived are taken in stride across all disciplines.

For engineers and product owners:

- Architectures are created by engineers with cost in mind from the beginning.
- Engineers include cost as a first-class metric, using it not only in build decisions, but also as an ongoing metric in their core KPIs to prioritize development work.
- FinOps certification and maturity metrics become a part of their job requirements, training programs, and incentive plans.
- Things that are scalable, elastic, or have dependencies on cloud services are all moved to cloud.
- Product teams consider and regularly look at cost in product pricing and other decisions.
- Engineering has a fully cloud-focused development platform with an integration pipeline and a DevOps approach. It is deploying resources only with automation and has a fully tagged environment, with consistent account and tag strategies tied to budgets aligned with business outcomes.

For finance:

- Finance has moved to an agile forecasting and budgeting process, empowered by all engineering teams who take responsibility for their own forecasts and budgets.
- Overhead on assets previously used in the data center are written down and eliminated.
- Finance provides ongoing input to operational allocation requirements like tag prerequisites that overlay existing accounting constructs to applications, and also access to data exports required for agile forecasting and budgeting.

For leadership:

- Management has fully embraced a cloud-based model of operating.
- FinOps is just part of the way the management expects the organization to use cloud effectively. Gone are the days of pitching the FinOps team.
- Leaders have clear visibility into the returns of their technology investments, line of sight to regular fluctuations in their forecasted spend, and a responsive path for providing strategic guidance on future investments.

As FinOps matures inside an organization, there's no clear line of where FinOps is happening and where it isn't. Everyone across the organization contributes in their own way to the success of value being derived from cloud usage.

In an ideal world FinOps just happens; everyone contributes the exact amount of effort needed with perfect alignment to the goal of getting the most value from the cloud. But we know that without someone coordinating all these efforts, the likelihood of this goal being achieved within an organization is low. Initially, the FinOps practitioner is employed to educate and build processes to enable and centrally manage elements of FinOps that are better operated at a single point within the organization. As the processes and education of FinOps mature across the organization, the potential for cloud wastage is considered by engineering groups *before* it is created, finance teams have active communications with engineering groups, expectations of cloud spend become aligned, and the amount of time and energy needed by the FinOps practitioner to keep FinOps operating successfully is reduced.

Think of the FinOps practitioner as a heater radiating out FinOps warmth. As the practitioner makes a positive impact across the organization they provide warmth to heat up others around them with motivation of doing FinOps well. These other personas then heat up as they contribute to FinOps and warm others around them. Eventually, the whole organization is warmed up and the amount of heat the FinOps practitioner needs to deliver is reduced.

The question then becomes if we take away the original source of the FinOps energy—the FinOps practitioner—will FinOps remain active within the organization?

Conclusion

You should now understand how the FinOps processes that you've built along the way enable you to begin measuring your cloud spend against the business value it delivers. You use the business value to set goals around optimizations and use metrics to track the true drivers of your cloud costs.

To summarize:

- The nirvana stage of FinOps is getting to a place where your organization makes ongoing, data-driven decisions on cloud spend based on business value.
- Who makes these decisions? Architects when they are designing infrastructure, engineers when they are writing code and deploying services, finance and procurement teams when they are making commitments to cloud providers, and leadership when they are driving technology strategy.
- It's an ongoing collaborative effort that can happen only when a FinOps cultural transformation has occurred.
- Your unit economic metrics measure the business value of your cloud spend and guide ongoing decisions related to it.
- Organizations with complex structures and multiple product offerings will often have multiple unit metrics, each one relevant to a specific product line, application, or service.
- Decisions are no longer focused on the cost of cloud, but are based on the benefits the cloud spend generates for the organization.

In the final chapter, we will close with talking about the engine behind FinOps: *you*.

CHAPTER 27
You Are the Secret Ingredient

To loosely quote the *Kung Fu Panda* movie, the secret ingredient to FinOps is that there is no secret ingredient to FinOps. The secret ingredient to success is you. And it's also every other person in your organization who influences cloud spend.

Spoiler alert: in the movie, the main character, Po (an endearingly clumsy panda) aims to be a great warrior. He receives a mystical scroll, which is supposed to unlock limitless power within whoever reads it. However, when he finally opens it, the scroll appears to be blank and has only a reflective surface inside. Distressed, Po gives up and returns to his boring life as a noodle soup vendor. Po asks his father the secret ingredient to the family's famous soup. His father makes a startling revelation: there is no secret ingredient to the soup—the secret ingredient is the people who make it. This is a perfect metaphor for FinOps. We can't tell you how to be a FinOps warrior, but we can tell you that if you lean into learning, collaborating, and improving, your practice—and your career—will flourish.

If you've read this far, you're probably either fairly far down the path of working in or alongside FinOps, or seriously considering a career in the field. While 60% of enterprises already have some level of FinOps practice in place as of 2022, most don't yet have a fully staffed team, and the other 40% will be staffing up quickly. Because if you're doing cloud at scale, you're going to need to do FinOps.

But here's the rub: FinOps needs people, and there simply are not enough FinOps practitioners out there to meet the demand. Yes, you can—and should—try to automate as many of the repetitive tasks as you can over time, but to grow, train, evangelize, and build the practice requires a FinOps team radiating best practices, standards, and patterns. As it develops, the practice needs thoughtful executives to put in top-down cultural changes, thoughtful engineering leaders to begin integrating cost into their team's workflows, thoughtful finance leaders to adopt more agile processes

to more successfully partner with their engineering counterparts, and everyone to realize that winning the cloud value game requires a team effort.

Call to Action

The biggest challenge facing the FinOps industry—and, one could argue, the cloud industry by extension—is a dearth of skilled FinOps practitioners. But they cannot be created overnight. Like any discipline, it takes years to become proficient at FinOps, and reading this book won't magically get you there—but it's a great start.

> *The best time to plant a tree was 20 years ago. The second best time is now.*
> —Chinese proverb

Our challenge to you is twofold. Go forward from here and commit to learning. But also begin contributing to the larger community. The first part helps you and your organization while the second one helps everyone, which in turn helps you out since a rising tide floats all boats.

The industry needs its practitioners to get more standardized, more rigorous, to share more best practices, to do open source code contribution, to contribute to training and encourage others to take it, to focus relentlessly on upskilling internal teams, and to look for opportunities to bring in fresh faces to the career path.

The FinOps Foundation offers training for various personas associated with the practice, but that is not enough. Ultimately, we need to work together to get cloud value learning fundamentals embedded into university programs, code schools, even high school computer science classes. Will there be a bachelor's degree in FinOps? Maybe not, but we believe a dedicated university module should be added to relevant degrees within the next few years.

The concepts you've read throughout these nearly 450 pages must start making their way back into conventional education programs not only for computer science–related disciplines, but also for accounting, business administration, finance, management, project management, and related degrees that ultimately end up being key stakeholders in a FinOps practice. Students graduating from any of these programs and heading to work in an enterprise that relies on cloud—and that is nearly every enterprise in the world—will be directly or indirectly involved in a FinOps practice.

There are norms for accounting, finance, and computer science that allow experienced practitioners in any of these fields to integrate quickly into similar roles at different companies. These same norms and principles, much of what we've seen discussed for years in the FinOps Foundation—and summarized in this book—must be carried forward to help make everyone more successful.

How to help:

- Read the evolving canon of open source practices developing on *FinOps.org*.
- Figure out where there are gaps that you can help fill: this raises the tide and it also gives your own career a visibility boost.
- Take FinOps training and give feedback on what else it needs to cover or what new trainings need to exist.
- Join a working group to collaborate with your peers on the definition of best practices.
- Listen in on—or offer to speak at—a FinOps event. There are virtual summits, local meetups, and larger in-person events happening all the time.
- Spread the word by bringing FinOps concepts into your work in related industry events and publications.

Need more ideas? See the FinOps Foundation Getting Started page for the latest ways to contribute back to the community, depending on your stage of maturity. You might start initially in listen-only mode but over time participate in working groups or even aspire to lead one, directly shaping the future of the industry.

The use of public cloud and its variable cost model will surely continue to grow. FinOps must grow with it to enable organizations to reap the benefits of using cloud without paying the enormous penalties that can result from using it poorly. You are a part of a vibrant and growing community that will be making the promise of digital transformation a reality for so many around the world, and we can't wait to see what you'll do next.

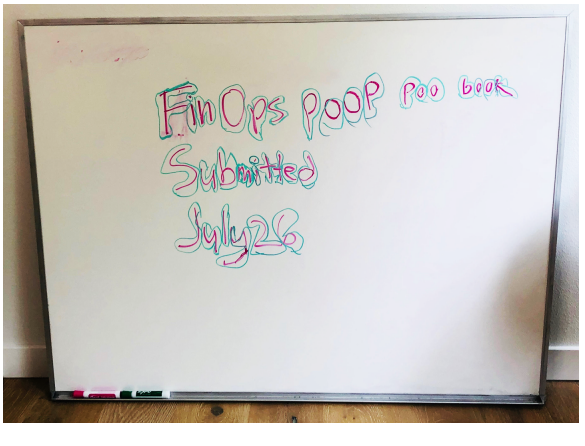
We look forward to seeing you in the community, where we can celebrate, struggle, learn, and build FinOps together.

Afterword on What to Prioritize (from J.R.)

Ten days after the deadline for the submission of the first full draft of the first edition of this book, one of my eight-year-old twin boys died. Two of his last three weekends I spent alone in my home office editing while my wife and the boys went out on adventures to places like the Oregon Coast.

On Friday, July 26, 2019, Mike and I had made that deadline. I felt proud. I commemorated the date by writing "FinOps Book Submitted July 26" on my whiteboard at home. I posted to LinkedIn about it being submitted. Mike, who had also spent countless evenings and weekends away from his family to write the book, also posted about the milestone.

The next day, Saturday morning, I should have been playing with my boys. But after months of writing (on top of my more than full-time day job), I was back in my home office to catch up on emails. My twins, Wiley and Oliver, came in playfully while I typed away. When I didn't pay enough attention to them, Wiley modified the aforementioned whiteboard text to read like this:



A little more than a week later, Wiley died in his sleep. Alongside the pain and grief was an overwhelming sense of regret for time with him lost to less important things. I wrote a long post on his life and death—and its impact on me. It went viral, I believe, because of its message of appreciating and prioritizing the time you have. I invite you to read it.

While this book is here to help your FinOps practice, don't let an expiring commitment or rightsizing sprint keep you from going home to be fully present with your family on Friday evening. Even if that commitment buy saves your company $100,000. It's not worth the lost time. Do it on Monday. If your boss doesn't like it, reach out to the FinOps Foundation group to find another job. It's a community of both experts and good people.

Your teams are unlikely to achieve long-term impact overnight. Find a good work–life balance. Even if your teams spend the weekend rightsizing everything, it's a one-time effort. Without building company-wide culture, processes, and practices that enable FinOps, these one-time projects on reducing costs will be short-lived. Weekends spent working will eventually be washed away in the sands of time.

And, finally, FinOps will evolve. It changes not only as cloud service providers continue to innovate, but also as more FinOps practitioners share their stories. This book is not the end of defining FinOps—it's really just the beginning. If you want to contribute to what's next, join the FinOps Foundation and be a part of the story.

Index

A

B

C

D

E

F

G

H

I

N

O

P

R

S

T

U

V

W

About the Authors

J.R. Storment is the Executive Director of the FinOps Foundation, a Linux Foundation program that boasts nearly 10,000 members worldwide and includes involvement from the majority of the world's largest cloud spenders. He was formerly the cofounder of Cloudability, a pioneer in the cloud cost management space from 2011 to 2019, when it was acquired by Apptio and became their flagship product. Post acquisition he spent a year as VP of Product/Engineering and GM of Cloud for the Cloudability business unit, before leaving to pursue his passion of advancing the people who work in the FinOps field as a full-time employee of the nonprofit Linux Foundation. As of 2023, he's 13 years into working closely with the largest cloud consumers in the world—from Apple to Spotify to BP to Nike to Uber to GE—helping them design strategies to optimize and analyze their cloud spend through technology, culture, and process. J.R. has spoken on the topics of FinOps at multiple AWS re:Invents and dozens of conferences across the US, APAC, UK, and EU. Born in Oregon and raised in Hawaii, J.R. has lived in San Francisco, Barcelona, and London and now resides back in Portland, Oregon, with his wife, Jessica, and son, Oliver. He identifies as a father of twin boys, though Oliver's brother Wiley passed away during the writing of the first edition of this book in 2019.

Mike Fuller is a principal engineer and has been working with Cloud and FinOps at Atlassian for over ten years. Since authoring the first edition of the *Cloud FinOps* book, Mike has formed a dedicated FinOps team within Atlassian, moving the roles and responsibilities out from the cloud center of excellence to form a dedicated team of data engineers, analysts, and FinOps practitioners. Mike's team helps Atlassian get the most value out of the money it spends on cloud. As a principal engineer within Atlassian, Mike is also an architect on the Insights team working on strategies to optimize the way domain experts (Security, Compliance, FinOps, Reliability) interact with the engineers across Atlassian's global workforce.

Mike holds a bachelor's degree in computer science from the University of Wollongong and has completed nine AWS certifications. He has presented at multiple AWS re:Invent and AWS Summit events on topics that include AWS security and Cloud FinOps. Mike has served as a member of the FinOps Foundation Technical Advisory Council and is currently a member of its Governing Board. Over the years Mike has made many friends across the globe who operate as FinOps leaders within their own organizations. These connections have helped shape Mike's thoughts on what it means to be successful at FinOps.

Mike lives on the south coast of Australia with his wife, Lesley, and two kids, Claire and Harrison. They split their family time between the beautiful beaches and rural landscapes that make Australia famous.

Colophon

The animal on the cover of *Cloud FinOps* is a blue-headed sunbird (*Cyanomitra alinae*). They are small birds that live along the African Great Lakes region in sub-Saharan Africa. As members of the family Nectariniidae, they have downward-curved bills and short wings that enable quick, direct flight. Being sexually dimorphic, the males sport a much more colorful plumage than the females; their feathers shine an iridescent bluish-green against bright red eyes. The sunbird can be found anywhere between the tropical rainforests of Nyungwe Forest National Park in Rwanda to the dense mountainous forests of Bwindi-Impenetrable National Park in southwestern Uganda.

The blue-headed sunbird spends much of the day along forest edges and clearings in search of food. It feeds on insects and small spiders, as well as nectar, just like the American hummingbird and the Australian honeyeater. Although unrelated to their New World counterparts, sunbirds do share a preference for feeding on red and orange tubular flowers, as well as having a brush-tipped tongue that allows them to extract the nectar.

Blue-headed sunbirds play a critical role in the African ecosystem because many of the continent's most iconic plants (such as aloes, proteas, and bird-of-paradise flowers) depend on it for pollination—so much so that it is widely thought that sunbirds have been a driving force in the plant speciation of Africa.

Many of the animals on O'Reilly covers are endangered; all of them are important to the world.

The cover illustration is by Karen Montgomery, based on a black and white engraving from *Wood's Illustrated Natural History*. The cover fonts are Gilroy Semibold and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.