O'REILLY®

Cloud FinOps

Collaborative, Real-Time Cloud Financial Management

J.R. Storment & Mike Fuller

O'REILLY®

Cloud FinOps

Applying traditional spend management processes to the cloud will lead to companies losing control of costs and interrupting innovation. Whether you're new to managing cloud spend or a seasoned pro, this book will clarify the often misunderstood workings of cloud billing. You'll learn expert strategies for creating a culture of cloud cost management.

Drawing on real-world examples of successes and failures of large-scale cloud spenders, this book outlines a road map for building a culture of Cloud FinOps in your organization. Beginning with fundamental cloud billing concepts, tech execs, finance teams, DevOps leaders, and FinOps practitioners will learn how to build an efficient and effective FinOps machine.

- Learn how the cloud works when it comes to financial management
- Set up a FinOps team and build a framework for making spend efficiency a priority
- Examine the anatomy of a cloud bill and learn how to manage it
- Get operational recipes for maximizing cloud efficiency
- Motivate engineering teams to take cost-saving actions
- Explore the FinOps lifecycle: inform, optimize, and operate
- Learn the DNA of a highly functional Cloud FinOps culture

J.R. Storment was the cofounder of Cloudability, the leading cloud FinOps platform for cost management and optimization. He's spent the last decade helping the largest public cloud consumers in the world design strategies to manage their cloud spend.

Mike Fuller is a principal systems engineer on Atlassian's cloud engineering team working on the design, governance, and implementation of best practices. Mike holds nine AWS certifications.

"FinOps has emerged as a new model to define the future of how finance and technical teams partner together. *Cloud FinOps* has provided a roadmap for those organizations looking to evolve the way they manage and optimize cloud expenditures, without slowing technical teams and innovation. 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

CLOUD COMPUTING / SYS ADMIN

US $59.99 CAN $79.99
ISBN: 978-1-492-05462-7

Twitter: @oreillymedia
facebook.com/oreilly

Praise for *Cloud FinOps*

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 roadmap 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

The freedom of cloud comes with the responsibility of FinOps. Businesses need to adopt an operating model in which software engineers are responsible for the cost of their solutions. This book covers the how-to's you need to get started.

—Dieter Matzion, FinOps Expert since 2013

So much about current cloud strategy is financial; the long-term business value of cloud-based solutions is predicated on making the right decisions about what needs to run where, according to the economics of cloud provision. However, the topic of cloud economics in general, and financial planning in particular, is poorly understood. This book cuts through the fog and gives cloud decision makers the clarity they need to make the right choices for their organizations, now and in preparation for the future.

—Jon Collins, VP of Research, GigaOm

Cloud FinOps

Collaborative, Real-Time Cloud Financial Management

J.R. Storment and Mike Fuller

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Cloud FinOps
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.

Acquisitions Editor: John Devins
Development Editor: Virginia Wilson
Production Editor: Kristen Brown
Copyeditor: Rachel Monaghan
Proofreader: Arthur Johnson

Indexer: Ellen Troutman-Zaig
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

December 2019: First Edition

Revision History for the First Edition
2019-12-12: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781492054627* for release details.

978-1-492-05462-7

[LSI]

For Wiley

Who dreamed of one day writing a book and waited patiently during two of the last three weekends of his life for us to finish this one. And who subsequently decided to retitle this book "FinOps Poop Poo Book."

Table of Contents

Preface

Over the years, we've heard the same stories over and over again. Engineering teams spend more than they need to on cloud, with little understanding of cost efficiency. Meanwhile, finance teams struggle to understand and keep up with what teams are spending. Then, to top it off, leadership doesn't have enough input into company spending—and sometimes doesn't even show a willingness to influence priorities.

Traditionally, procurement was largely in control of any material IT spend because they had to approve all large equipment purchases. As organizations move into the cloud, however, the pay-as-you-go model—also known as the *variable spend model*—allows engineers to bypass this procurement approval process. When cloud spend reaches material levels, organizations are left to allocate, explain, and control these costs. With over 300,000 SKUs offered by cloud service providers, and thousands of new features introduced each year, cloud financial management is a problem that isn't going to solve itself.

We need a new cloud financial management operating model. Enter FinOps.

The success stories we typically hear at conferences or read in blogs focus on how organizations have migrated their technology. They play up the challenges their development and operations teams faced and finally overcame in heroic fashion. They talk about scale and data, and the newest service they used to solve a complex problem. Often overlooked, however, are the financial management practices that enabled these accomplishments. Throughout the years, we've seen many cloud stories fall into trouble due to ineffective cloud financial management.

During the last eight years of our respective careers, we've heard a consistent theme from practitioners and executives alike: there's a lack of FinOps 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 Apptio Cloudability's cloud spend management platform.

Enterprises and tech unicorns alike struggle to evolve the way their teams work in the world of DevOps plus cloud. As they try to codify day-to-day best practices, they end up reinventing the same wheel, and they don't have a broad peer group to which they can turn.

But the wisdom is out there. The few enterprises further along the maturity curve (call them the FinOps Heroes) have broken down the silos. They get huge savings over what they would have paid in a pre-FinOps world. Meanwhile, their engineers are delivering innovation at faster speeds. Procurement has shifted to strategic sourcing and owning the relationship with the cloud provider. No longer removed from the process, the finance team is a proactive partner who has become technically enabled and is focusing on unit economics. And leadership is making intentional and frequent strategic choices among speed, quality, and cost.

We've heard an oft-repeated demand for a resource from which we can all learn, along with a need for someone to both formally define FinOps and draw from the great FinOps practitioners in the industry. In short, we want to capture what it is that makes them successful and share it with the rest of the world.

It's why we formed the FinOps Foundation (*http://finops.org*). It's that organization's practitioners who have fueled the best practices we'll cover in this book. All the examples we describe have been created out of their feedback. 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 and operations teams are most likely not used to thinking about costs as a day-to-day concern. In the old precloud days, they worried about performance. Constrained by hardware procurement and unable to get more servers whenever they needed them, they had to hoard resources or plan ahead. Capacity planning was done months, if not years, in advance. Now, in the cloud, they can throw company dollars at the problem whenever extra capacity is required. But that adds a new dimension to the work. They also have to think about the cost of their infrastructure choices and its impact on the business. At first, this feels foreign and at odds with the primary focus of shipping features. Then they quickly realize that cost is just another efficiency metric 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 teams 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 Capex reporting to transparent and fluid OpEx forecasting. As part of that role, finance teams become 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.

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

Tech executives, such as a CIO or CTO, have likely lost control of many spending decisions and now must trust teams to operate within reasonable budgets. Tech executives no longer plan large purchase decisions. Instead, they think more about how to forecast spend that's currently 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. Tech executives 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 working with some of the most experienced cloud financial management teams—teams who manage hundreds of millions of dollars 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 it fits into an 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 are looking to change that. Our founding charter members include some of the greatest FinOps minds from such diverse companies as Spotify, Nationwide, Pearson, Atlassian, Thermo Fisher Scientific, HERE Technologies, and Australia Post. The FinOps Foundation is a nonprofit trade association focused on codifying and promoting cloud financial management best practices and standards.

After you've read this book, the FinOps Foundation is a great place to go to continue learning about FinOps. It's also a vibrant community where practitioners meet to discuss war stories online or in virtual meetings.

We hope that the real-world strategies, processes, and stories in this book will inspire us all to better take control of our cloud spend. And in the process, we can make our companies, and perhaps our careers, more competitive. Anyone who believes their company and career need to shift to a new, more effective means of cloud financial management is welcome to join the FinOps Foundation.

We use Apptio Cloudability as an example of a cloud financial management platform throughout this book, as J.R. was cofounder of Cloudability and has been through the whole cloud financial management maturity curve since the beginning. However, a specific technology platform isn't required for FinOps. You may be accomplishing some of these best practices using other tools, open source, or scripts you've built yourself.

What You Need to Know Before Reading On

We're going to share how we can all thrive in the new world of FinOps and, in so doing, help our companies to become more competitive. 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 Fundamentals 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), and Spot. It's OK if you don't know how RIs 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 EC2 resources.

FinOps Is Evolving

Over the past few years, what we call FinOps today has been evolving—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 people challenging the way things are done to help formulate the successful FinOps practices we have today.

Check out *http://finops.org/book* to stay up to date and for responses to any recent changes in cloud.

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.

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, conferences, 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, please visit *http://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/cloudFinOps*.

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

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

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

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

First, to our families, who sacrificed so many nights and weekends in allowing us to write this book, and who suffered through us talking about nothing else but FinOps and the content we've built: thank you.

To the team at O'Reilly (Amelia Blevins, John Devins, Sarah Grey, and Virginia Wilson): without your efforts this would never have happened.

To Gavin Cahill, who stepped in to spend countless hours at the end of the process polishing all of our rough edges: thank you for your massive efforts to bring the book over the finish line.

Thanks to Ken Boynton of Message Glue for polishing and clarifying our FinOps story to give it a more cohesive voice.

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

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.

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 our technical book reviewers and all the people who reviewed our content along the way; without your feedback this book would not be what it is today: Adam Heher, Alex Hullah, Alex Landis, Alexander Price, Alex Sung, Alison Pumma, Ally Anderson, Anders Hagman, Andrew Midgley, Andrew Thornberry, Anthony Tambasco, Ashley Hromatko, Ben Kwan, Bhupendra Hirani, Bob Nemeth, Casey Doran, Dana Martin, Darek Gajewski, David Andrews, David Angot, David Arnold, David Shurtliff, David Vale, Dean Layton-James, Dieter Matzion, Elliot Borst, Elliott Spira, Ephraim Baron, Erik Onnen, James Jackson, Jason Fuller, Jess Belliveau, John Merritt, John Stuart, Jon Collins, Joseph Daly, Justin Kean, Keith Jarrett, Manish Dalwadi, Marsha Shoemaker, Martin Bleakley, Matt Finlayson, Matt Leonard, Michele Allesandrini, Nabil Zakaria, Naveen Chand, Pedro Silva, Phillip Coletti, Rob Martin, Sascha Curth, Shane Anderson, Stephanie Gooch, Stephen Elliot, Tom March, Tom Marrs, Tony Curry, Umang Sehgal, Vasilio Markanastasakis, and Wendy Smith.

PART I
Introducing FinOps

In the first part of this book, we cover a lot of fundamentals, such as what FinOps is, how teams operate, the language of FinOps, and how cloud services are billed.

CHAPTER 1

What Is FinOps?

FinOps is a direct response to the stimulus of cloud. It's not something that one person or company came up with; it's a movement that evolved spontaneously, around the world, due to the environment created by on-demand cloud resources encountering no-longer-relevant technology management methodology. It is a cultural change and set of processes that has been—and likely will be—called other names.

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 cloud moves ownership of technology and financial decision making out to the edges of the organization. It flips long-held, forward-looking capacity planning methodology on its head to become rate-optimization analysis for technology that's already been used. And it forces IT, finance, and business professionals to work together in unfamiliar ways. It's an acceptance that the old ways of managing infrastructure aren't just ineffective in cloud; they are irrelevant.

In this chapter, we'll describe the core principles of FinOps, where the movement came from, and why every organization needs to embrace it for cloud success.

But first, let's set the stage for defining FinOps with a 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 IT and virtualized servers. Here's a typical story 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 determining the production needs of the organization to meet changing demands for its products. There weren't a lot of surprises.

Then Finn notices an increasing number of AWS or GCP payables coming in without purchase orders attached. One of his cloud-savvy colleagues, Sarah, steps up and explains the highly variable nature of cloud and how it's just, well, different than on-premise—and that there's an entirely new way of managing it, as well as a new professional discipline emerging to do so.

Finn carefully considers Sarah's words. It does sound interesting, and appealing. But then Finn remembers how well the processes he has in place for the organization's 8,000 on-premise servers work. Cloud couldn't be that different.

The next quarter, cloud spending doubles unexpectedly, and Finn goes back to Sarah with his tail between his legs. He commits to trying a new set of processes that look more frequently at spend and increasing his interface with the teams creating the spend.

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

It's go time. Finn rolls 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's a path forward that seems to allow him to account for the spend and the teams to get the tech they need without friction. Cloud migration begins to soar.

Right when things seem to be going well, a new CIO comes in and says the cloud is too expensive at scale, advocating for a widespread return to the data center. It's time for Finn's biggest test: working with his cloud-savvy colleagues to show that cloud is more than a cost center. They must show it can enable innovation and velocity in ways that on-premise cannot, driving competitive advantage for the company. The CEO sees the bigger picture and agrees, paving the way for a cloud-first strategy.

Newly confident, Finn forges ahead, breaking down silos between teams and helping to drive real change in the organization. But he faces one last battle: cloud spend is now hitting materials levels, affecting the bottom line. The CFO steps in to stop the upward trend through any means necessary and ensure margins are not affected. Finn must move beyond the one-dimensional view of looking at cloud spend only and shift to a unit

economics model that ties the spend back to business value, giving him the ammunition to show how to ensure, once and for all, that cloud spend is on the right path.

In the end, Finn realizes that this journey is just the beginning. Cloud is evolving; FinOps is evolving. Finn and Sarah 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, gave us our first glimpse into what would become FinOps around 2012 in San Francisco. A couple of years later, we saw forward-looking enterprises in Australia, like Qantas and Tabcorp, begin a similar practice. Then, in 2017, during J.R.'s two-year tour of duty in London, he was a firsthand witness to enterprises like BP and Sainsbury's as they developed this new approach across their cultures. FinOps came into being slowly, all over the world, as the financial and accountability challenges of cloud presented themselves at scale in each territory.

"FinOps" is a term that has come late to the party. In the early days, companies were simply calling 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 was eventually replaced by "FinOps." 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 being adopted worldwide. Recently, Nike posted a director-level job role in their newly formed Cloud Business Office. Southwest Airlines formed a FinOps Business Office. And job listings for FinOps managers at big cloud enterprises like Pearson, Spotify, and Randstadt are popping up more frequently on LinkedIn. Tech unicorns like Atlassian have begun operationalizing FinOps through their team collaboration software.

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.

The Definition

FinOps is the practice of bringing financial accountability to the variable spend model of cloud, enabling distributed teams to make business trade-offs between speed, cost, and quality.

At its core, FinOps is a cultural practice. It's the most efficient way in the world 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 work together to enable faster delivery, while at the same time gaining more financial and operational control.

No longer is a siloed procurement team identifying costs and signing off on them. Instead, a cross-functional FinOps team adopts a definitive series of procurement best practices, enabling them to pull together technology, business, and finance in order to optimize cloud vendor management, rate, and discounting.

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*.

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 intelligent decisions that ultimately 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*

If it seems that 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.

Jason Fuller, who runs FinOps at HERE Technologies, 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.

Real-Time Reporting (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, the 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.

This real-time visual cue typically creates 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 during his talk at the 2019 Google Cloud Next event[2]

This real-time data-driven decision enablement is what FinOps delivers. In the data center world, engineers take individual actions that can't easily be traced to their financial impact on the company. With cloud, it's possible to gather that data, but simply being in cloud doesn't automatically give you that feedback loop.

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 couldn't implement a feedback loop.

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 dev environments that they really didn't need. With only three hours

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*.

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 understand just how large the business impact was. This early FinOps process turned cost into another efficiency metric to consider. And anyone who knows engineers knows that they tend to dislike inefficiency.

FinOps is about helping the business make better decisions while moving more quickly. And it achieves this by enabling frictionless conversations between teams—the source of increased velocity.

> 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: A Novel About IT, DevOps, and Helping Your Business Win*

Teams that previously spoke different languages and kept each other at arm's length now build frictionless 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, businesses enable productive trade-off conversations to happen—even when other teams are not a part of them. We will cover this common lexicon in Chapter 4.

Core Principles of FinOps

We believe having values of FinOps 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.

- 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 economics 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.
  - Individual feature and product teams are empowered to manage their own usage of cloud against their budget.
  - Decentralize the decision making about resource usage and optimization.
  - Technical teams must begin to consider cost as a new efficiency metric.
- FinOps reports should be accessible and timely.
  - Process cost data as soon as it becomes available.
  - Visibility 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.
  - Centralized automation for FinOps reduces duplicated effort.
  - Executive buy-in for FinOps and its practices and processes is required.
  - Rate and discount optimization is centralized.
  - Centrally govern and control Committed Use Discounts, Reserved Instances, and Volume/Custom Discounts with cloud providers.
  - Remove the need for engineers and operations teams to think about rate negotiations; then, stay focused on usage optimization.
- Take advantage of the variable cost model of the cloud.
  - The variable cost model of the cloud should be viewed as an opportunity, not a risk.
  - Embrace just-in-time prediction, planning, and purchasing of capacity.

— Agile iterative planning is preferred over static long-term plans.

— Make continuous small adjustments in cloud usage/optimization.

When Should You Start FinOps?

Due to the sheer number of blogs, talks, and sales pitches that focus heavily on cost optimization, many people believe that the right time to implement FinOps should be measured by the amount of their cloud spend. And this does make sense on some levels. For example, a massive cloud spender could immediately find a lot of potential savings. However, experience has shown that a single well-managed team spending millions on the cloud might actually benefit less from FinOps than an organization with many teams that have smaller cloud deployments.

A successful FinOps practice doesn't require sizeable cloud deployments or a multimillion-dollar cloud bill. Starting FinOps sooner will make it much easier for an organization to make more informed decisions about cloud spend, even as its operations scale. Therefore, it is essential to understand a FinOps concept we call *Crawl, Walk, Run*.

Although this book, and the FinOps Foundation, can guide an organization to successful practices while helping teams avoid common pitfalls, no organization can proceed directly from zero FinOps to perfectly 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, ultimately, a cultural shift, and the earlier it starts, the sooner an organization can achieve successful cloud cost management.

Our experience has taught us that you do FinOps from day one, but you engage more of the processes as you scale up. There are typically two times when companies implement FinOps:

- The most common is when things go 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 the most 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 less common (and wiser) approach is taken by executives who have seen the cloud journey play out and aligns to the Crawl, Walk, Run model. The practice needs to develop at a pace that matches the company's position in the FinOps maturity cycle. For example, 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.

But no matter how a company arrives at the decision to implement FinOps, the first critical step is to get visibility in a near-real-time manner, so everyone can see what's happening and catch cost overruns before they become too large. While that visibility is being achieved, the FinOps teams start educating the entire business. As cross-functional teams work together, finance people will learn more of the language of cloud, while engineers begin to grasp financial concepts.

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: Unit Economics

One of the most important concepts in FinOps is *unit economics*. The idea is to measure cloud spend against a business metric (total revenue, shipments made, paid subscribers, customer orders completed, etc.). Choosing the right business metric is a complex process, one we'll cover in Chapter 19. For now, the main thing to remember is that unit economics relies on almost every aspect of FinOps, including tagging, cost allocation, cost optimization, 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 for the organization. Then, as services evolve or change entirely with new features, companies are able to measure the impact of these changes via these business metrics.

The result is companies that can determine the difference between good cloud spend and bad—and that trend those decisions over time. You should keep business metrics in mind throughout the book and as you implement FinOps inside an organization. As your practice matures, you will be able to implement and see the value of unit economics.

Conclusion

In this chapter we've codified the core principles and values that can guide any organization that wishes to start a FinOps practice. Those organizations can now align to what FinOps is intended to achieve and follow a series of best practices to ensure their success as they Crawl, Walk, and Run along their FinOps journeys.

To summarize:

- FinOps is about collaboration between all teams inside an organization.
- Everyone has a part to play and should become cost-aware.
- The core principles of FinOps should be the foundation of all processes around cloud financial management.
- Real-time reporting gauges your current spend and optimizations.
- Data-driven processes are key to an organization becoming cost-efficient.
- Business decisions can accelerate and match the rate of cloud resource decisions.

Now that you have an understanding of what FinOps is, let's look at what the cloud enables inside organizations and at why you should 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.

Use Cloud for the Right Reasons

Savings are often touted as the primary benefit of 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.

Running in the cloud also gives enterprises the ability to move faster and grow their revenue. Even businesses in "nontech" sectors, like airlines, banks, and retail companies, are turning to software and data to differentiate themselves from competitors. Software helps connect them 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]

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

In 2018, 7 of the 10 most valuable companies in the world (market cap) were software-first focused. Tech titans such as Microsoft, Amazon, Alibaba, and Tencent sit alongside banking stalwarts like JPMorgan Chase and Bank of America, all using software as the primary way of connecting to their customers. Even traditionally non-technology 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 speeds never before possible.

Cloud is now central to the operations organizations provide customers. Racking servers and engineering "solved problems" from the ground up isn't how companies differentiate themselves anymore. Cloud has made the latest technology—scalable infrastructure, machine learning, or IoT (Internet of Things)—available on-demand for businesses of all sizes. Traditional enterprises struggle to compete against the technology scale of Google, AWS, and Microsoft in running infrastructure and attracting the talent to support it.

Enterprises that win align their organizations around empowering their engineers to write apps better and faster. And those engineers can increasingly get whatever they want, when they need it. They can focus less on infrastructure and more on apps. The cloud has fundamentally transformed their ability to ship more competitive products to their customers. As a result, companies look to move faster and get better every day instead of every quarter or so.

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

The Problem

Cloud spend has begun to hit a tipping point. The latest Gartner forecast puts it at $360 billion by 2022. Our guess? The actual number will end up even higher. In our experience, organizations underestimate their cloud spend and industry forecasts are regularly revised upward. Cloud spend is becoming a material part of organizational budgets, impacting both the bottom and top lines for enterprise P&Ls. The most scaled spenders are already into the nine-figure-per-year territory, some with the potential to hit $500 million or more per year in IaaS spending.

And with all that growth, the time-honored silos between tech, finance, and procurement have become a problem. To understand why, let's look to our favorite definition

of cloud, by Dave Nielsen back in 2011. In order to be called "cloud," Dave said, an infrastructure provider had to be "OSSM":

- *O*n-demand
- Scalable
- Self-service
- *M*easurable

In the public cloud, the "OSS" in OSSM is what drives incredible innovation while at the same time creating the need for a new operating model. 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 processes ineffective. Engineers aren't getting approval from the central procurement teams for each individual cloud resource, and the idea of trying to implement some micro-approval process threatens to slow innovation, one of the major benefits of cloud.

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

- Equipment was often overspec, 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 upfront 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 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.

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

The Impact of Not Adopting FinOps

When you're focused on speed and innovation, it's easy for cloud bills to soar. In turn, companies clamp down on spend, innovation slows, and there's the looming danger of companies becoming less competitive. This is where FinOps enters the picture. In the same way that DevOps revolutionized development by breaking down silos and increasing agility, FinOps increases the business value of cloud by bringing together technology, business, and finance professionals with a new set of processes and culture.

Even though cloud spending is now a material amount of IT spend, the cloud operating model is still immature. As companies move away from the fixed-cost data center model and toward the cloud with its variable-cost, consumption-based model, they must transform their financial operations, just as they did their development ones.

Companies must bid farewell to a world where only a few people make buying decisions once a quarter or once a year. The three- to five-year investment cycle must also be let go. Decision making in the cloud should, by necessity, be distributed across technology, business, and finance teams. Doing so makes logical sense, but organizations often struggle to balance operational and financial control with the new speed of high-velocity decision making. But you seek that balance with FinOps. Otherwise, you'll see one or both of these undesired outcomes: cloud-bloat inefficiencies slowing down the business, and sticker shock over the monthly cloud bill.

The good news is that you can have your cake and eat it, too. FinOps processes enable these cross-functional teams to operate at high velocity while improving the unit economics of cloud. The move to variable spending in cloud, coupled with the need for distributed decision making, coaxes technology teams to partner with finance and business teams to make informed decisions so their organizations can continue to drive innovation at velocity.

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.
- The procurement team no longer has control of the spending. In cloud this power has been pushed to engineers.
- FinOps allows you to operate at the per-second speed of cloud rather than relying on traditional monthly or quarterly spend reviews, which allows you to avoid unexpected costs.

FinOps is brand new to most companies and must be implemented and adopted throughout the entire organization to be successful. In the next chapter, we'll examine the culture of FinOps and the roles everyone must play in order to enable a successful FinOps practice inside any 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 approaching the cloud and cloud financial management. Technology certainly helps solve problems like checking reservation coverage 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 sit in the 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 or solved by the people and teams across the organization. Ultimately, it comes down to you. Will your teams have the right values and traits for successful FinOps?

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.

> Everyone's job is 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, albeit at different levels of required tooling and processes.

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 and prepurchases to cloud service providers. While these will undoubtedly save the organization money, trust is required between 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 try to explain their way out of making necessary performance modifications. Expertise builds trust and credibility, and it's the only way to build a strong FinOps foundation.

FinOps practitioners should have the autonomy to share impartial best practices while being fully supported by executive buy-in. 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 team is the driver of a FinOps mindset across the organization, evangelizing best practices for incorporating cost considerations into development plans. FinOps practitioners perform centralized tasks like managing and reporting on reserved cloud capacity. But the actual implementation of those best practices is handled by the engineers inside various product and application teams. A good example of this is rightsizing and idle usage reduction. A FinOps team will set the goals, but engineers will be the ones changing out the instances and building start/stop parameters in the code. And since the FinOps team is spending their time on these best practices and efforts, the central team is free to focus on not slowing down the pace of innovation.

> 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, Director of Cloud Cost 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 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 8.

Why a Centralized Team?

The unbiased central team, with ties to both 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.

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. On the other hand, if cost allocation is done independently by each team, there is often 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. A centralized 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 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.

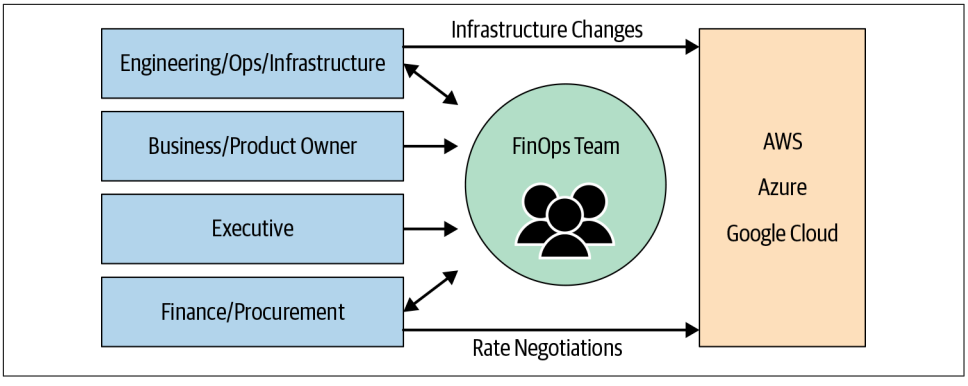


Figure 3-1. Team interaction around FinOps

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

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, which we'll discuss in Chapter 19.

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're often tech-savvy financial analysts with titles like Cloud Cost Optimization Manager, Cloud FinOps Analyst, Director of Cloud Optimization, Manager of Cloud Operations, or Cloud Cost Optimization Data Analyst.

Engineering and operations

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 and procurement/sourcing

Finance and procurement team members, including Technology Procurement Manager, Global Technology Procurement, Financial Planning and Analyst Manager, and Financial Business Advisor, 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.

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 executive 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. With FinOps, you seek a balance among speed, cost, and quality—for example, "Move fast, but keep our growth in spending below 5% per month."

Where Does Your FinOps Team Sit?

FinOps teams don't yet have a default position within the organizational hierarchy. When we posed the above question to the FinOps Foundation charter members, we found they had each placed their FinOps team in a slightly different location within their organization.

Most commonly, the FinOps team is aligned within or alongside the tech operations teams. This means it is typically part of the CTO or Head of Technology function. One member found this 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. Also, tech operations is perhaps the easiest place to start the team, if only because they are the ones who best understand the complexities of cloud.

After observing so many internal FinOps teams, we think the right place for FinOps is in business operations, under a Chief Operating Officer (COO). This allows the FinOps team to align the day-to-day operational processes of FinOps to the company's goals and strategies, as set by the CEO. For example, if the company is tightening to increase margins, the FinOps team may focus their efforts on reducing waste. If the company is focused on growth and capturing market share, they may focus on enabling velocity above all else. Sitting in this position, FinOps creates faster decisions among good, fast, and cheap.

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

In Figure 3-2, the FinOps practitioner reports to both a technology leader and a finance leader and partners with automation and optimization engineers to enact change in the organization. Their role is focused on financial analysis, monitoring, tracking spend, and optimization efforts. The automation manager helps implement and automate actions like rightsizing.

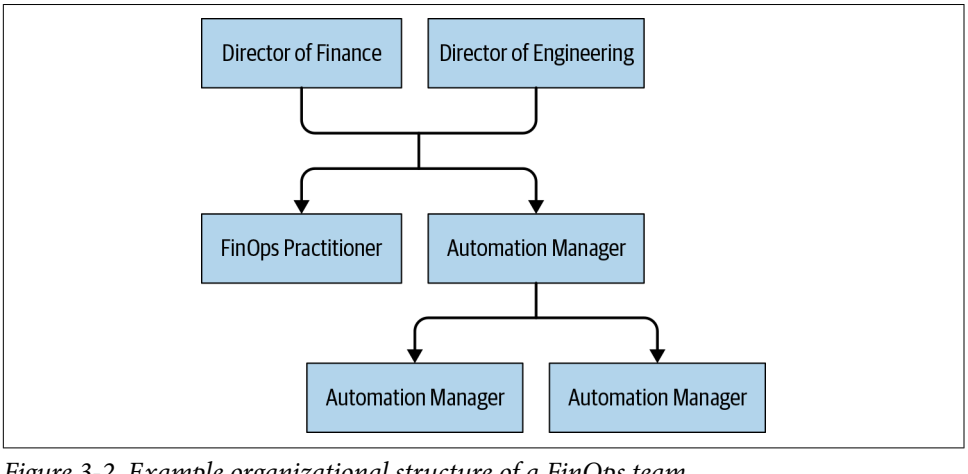


Figure 3-2. Example organizational structure of a FinOps team

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 different personas.

Engineers

- Want to work on something meaningful and fun
- Want to deliver software fast and reliably
- Hate inefficiency and want efficient use of resources
- Stay up to speed on the latest tech
- Are measured by uptime
- Want to deliver features, fix bugs, and improve performance
- Would prefer not to worry about cost (but are responsible for incurring it)

Usually, engineers focus on getting the latest software features or bug fixes to customers instead of on the finance side of things.

Financial operations 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 Management 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."

Ideally, a FinOps practitioner will spend more time performing these tasks on behalf of engineers, choosing to make recommendations for where engineers are able to optimize and then helping them by removing—and standardizing—processes that can be centralized. This approach allows engineers to focus as much as possible on their services.

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

Finance people can experience sticker shock when moving from capital expense to operational expense. They're used to capital and its three-year depreciation, but cloud spend is a lot more fluid. They're often not sure how to keep up with the rate of change and don't know if they can trust the numbers.

Thus it's important to remind traditional finance teams that cloud fits a model they already understand. It's simply an operating expense. It just happens to move in microseconds, and spend can go out of control over 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

- 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

In Chapter 2, we 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, teams work out the balance among good, fast, and cheap.

Procurement and Sourcing People

- Traditionally wanted to measure on savings or cost avoidance
- Want to get the chosen technology in the hands of teams quickly
- Wish to retain some measure of control over spending
- 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 standing 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 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. Alternately, 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.

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 as part of each job description or building out FinOps learning as part of onboarding your new hires into the organization. Ask cloud operations engineers how cost metrics play a part in good service design, and ask finance members 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 technologists.

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

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

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

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

FinOps practitioner
: Focuses on cloud spend, staying on budget, cost aversion, and cost savings.

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 containers.

You might think that having more services on the same compute resources would mean more efficient costs, but those costs come with a loss of visibility. Billing data has the underlying compute instance costs, but there's nothing to help finance work out what containers 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. As 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.

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.

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 and Cloud

Successful FinOps involves teams from multiple parts of the business collaborating. Each team within your organization uses domain-specific terms, views cloud costs from a different perspective, and has separate motivators for what they would like to achieve. In this chapter, we discuss how you can enable teams to collaborate effectively by educating staff both on specific terms being used and on methods for avoiding confusing language in order to get a point across.

Stories from the Cloud—Mike

Prior to starting my FinOps practice, 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.

Granted, this is a story of a Run stage company. A Walk stage company 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.

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 uses terms like *usage*, *rates*, *costs*, and possibly *utilization*. Engineers, on the other hand, refer to *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 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, asked procurement what spend was committed and which budgets it should be depreciated against.

When you distribute the purchase approvals 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. The language gets more complex when we add in cloud service provider–specific terms, and it quickly approaches incoherence as we 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
: The process of splitting up a cloud bill and associating the costs to each cost center. We'll look more closely at this process in Chapter 9. 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.

Rightsizing
: When a cloud resource is provisioned larger than is required, such as having too much memory or compute power, it's considered oversized. Rightsizing is the act of changing the size of provisioned resources to one that better matches needs.

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 Reserved Instances (RIs), Committed Use Discounts (CUDs), or commercial agreements between an organization and a cloud service provider in order 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 11. 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.

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.

Reservations/commitments
: By precommitting to a cloud service provider a set amount of resource usage using RIs or CUDs, you receive a reduction in the rate normally paid for those resources.

Reservations 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*.

Reservation 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 reservation 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 reservation. If you have resource usage that spikes during business hours and then is removed after business hours, committing to a reservation would result in reservation wastage and wouldn't save money. When usage would result in savings by being covered with a reservation, classify it as coverable.

Unblended rates
: Some resources are charged in decreasing rates the more you use them. (We'll cover volume discounts and sustained use discounts in Part III of the book.) 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.

Blended rates
: Some cloud service providers offer 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. While some cloud service providers offer a blended rate in their detailed billing data, often the way the costs are blended is not perfectly even, or some resource costs are not blended, which can lead to confusion about the true cost of a resource.

Amortized costs
: Some cloud resources and reservations come with an upfront fee. The amortized cost of a resource takes this initial payment into account and divides it out, attributing the prorated cost for each hour of billing.

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.

Defining Finance Terms for Cloud Professionals

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, this means you should expense spending using the billing data (e.g., Cost and Usage Reports in AWS, Cloud Billing Reports in GCP, Azure Billing File in Azure) rather than using the invoices from the provider.

Capitalized expense (CapEx) versus operational expense (OpEx)
: When you capitalize something, it becomes an asset of the company, whether or not it gets expensed within a specific period. The test you can apply is: if an organization writes a check to acquire something, does that acquisition benefit future periods? If it does, then it can be capitalized. If it benefits only the current period, then it's an expense that is expended in this period with no future benefit, making it an operational expense. Capitalization causes total outlays to differ from expenses in a similar period, with the delta being that which is capitalized.

Cost of capital/WACC
: Cost of capital refers to the cost to an enterprise to deploy their money toward an investment. In cloud, cost of capital is an important consideration when looking at 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 RI purchases.

Cost of goods sold (COGS)

COGS measures how many dollars of outlay it takes to generate revenues in a specific period. 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?

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. You can't usually materially turn down your sales commissions or fire your support people, which 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 for future periods, the cost of the coal would then be 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.

How do COGS and capitalization come together in cloud?

We recently saw a UK retailer that was developing a new shopping platform product and was applying these principles in an interesting way.

The company accounted for the EC2 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 EC2 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 EC2 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.

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 we would get for $10 is relatively simple. However, if a resource costs $0.0034 per hour, working out how many hours we 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.

In a FinOps Foundation meeting, one of our 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.

We've discovered that 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 in 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 it worth putting in to achieve the savings?

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 go against the common lexicon. Sometimes the specific number of dollars spent or saved isn't 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.

That same FinOps Foundation member correlated the amount of cost savings to cases of beer. If the team took a certain set of actions, they found they could save the equivalent of tens of thousands of beers. It may not surprise anyone that using beer as a unit of measurement was more clearly understood by engineers than raw dollar values.

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 business leaders, seeing cloud costs as a percentage of the company's total revenue gives a complete 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. We'll cover optimization methods in Part III of this book.

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. For a transport company, your unit metric might be packages delivered, or you might start with a simpler metric like percentage of revenue spent on cloud. 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 server instances. We'll cover these in more detail in Chapter 13, but generally the idea is that if you make a commitment to a cloud service provider and use those reservations effectively, you can save money. If you don't, you lose money. AWS calls its offering Reserved Instances, with each size server saving a different amount.

If your FinOps team creates a report for themselves, they need the details on the individual reservations made. But as you start reporting up to finance and then onto 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 *RI 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 further 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 Babel Fish Between Your DevOps and Finance Teams

We've been in meetings where an operations 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 would have needed a translator—or the aforementioned fish—to cut through the confusion.

But if both the finance and operations 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 fish 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 Both Sides of the House

As noted, a FinOps team should help define the terms used by an organization to describe cloud spend and savings. Both industry-standard terms and variations used by different cloud service providers must be adopted, and/or adapted, into the common lexicon.

No one in finance or operations should be expected to try to learn all the common language of the other team. Ideally, everyone meets in the middle, where finance learns the necessary terms used to describe cloud resources and operations learns the terms used to describe costs. 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. We'll dig deeper into the metrics used to compare groups in Chapter 16, but for now think of how being able to compare teams can lead into *gamification*.

For example, badges can be awarded for successful management of team cloud spend. Awarding 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" when we get to practices that help us reach our 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 domain-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.
- Moving away from reporting only in terms of dollars and cents to abstracted units of measurements will allow you to build reports 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, we'll look into the nuances of how companies are charged for cloud resources. Correctly understanding this will help FinOps practitioners to create valid reporting that will assist all teams in making sense of the cloud bill.

Cloud Billing Complexity

Cloud billing data is complex. AWS alone has over 200,000 individual product SKUs, some that are billed at per-second resolution. This data 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.

While there are platforms on the market 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 a FinOps platform.

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 a well-architected FinOps platform will help align the two for

invoice reconciliation each month, the levels of granularity with which invoices apply amortizations, blended rates, or Committed Use Discounts may still vary from detailed billing data at the service or account/subscription 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 cloud spend.

To identify cost drivers of any service, we need to get below the monthly detail and cloud service level provided by the invoices. AWS, GCP, and Azure all offer solid entry-level cost tools to analyze spend at a high level, and those help us take that first important step into visibility. However, these tools will often hit a wall once an organization goes multicloud, has custom negotiated rates, needs to push spend data out to create team accountability, or is getting into container cost allocation.

The Basic Format of the 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:

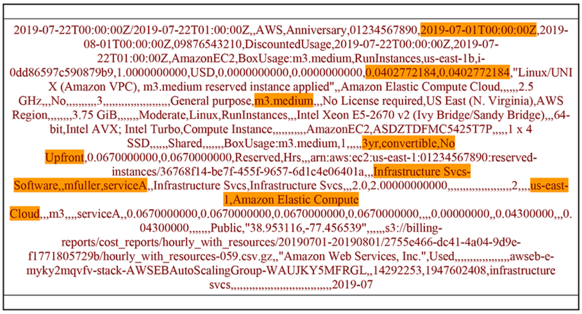
- Time period
- Amount of the resource used
- Rate details used for the charge during that period
- Where in the world it is located by region
- The resource ID
- Metadata attributes—like account or project—that can be used to allocate the spend to a product

Later, in Chapter 9, we'll cover 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. On the surface this appears incoherent, but you can discern a number of important attributes, such as:

- 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/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/UNIX (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 this granularity creates complexity, and eventually billing data gets so large that it can't be managed with spreadsheets. You must turn to computers for help.

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 million/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.

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."

Given that, Hootie is unlikely to get involved in FinOps, because cloud billing is all about time. Every charge has a time component. 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.

There are, of course, a few exceptions to this rule. Serverless and data transfer are not time-based but rather are 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 you did, then you're charged. If you didn't use the thing, then you're not charged. It's another example of the variable spend model of cloud versus the fixed spend model of on-premise.

Consider how an instance/VM/compute is charged by the cloud provider. The charge is based on how long the resource ran. Although most of the 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 in a minute).

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 a Reserved Instance 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, and the other would list the 200 hours under the RI.

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 to companies this way: "Don't get attached. You're not buying *things*—you're buying proportions of usage of things over time." That's an unwieldy sentence that gradually becomes crystal clear. Back in the '90s, Keck Observatory in Hawaii named each of its servers after a local beach: *Hapuna*, *Kiholo*, *Kaunaoa*, *Maumae*. But now, compute resources aren't individually delineated.

As we'll explain further later on, even RIs don't pay attention to what server they're attached to. They simply attach to one that matches their specific attributes. 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, say, AWS's 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 or *Crawl, Walk, Run* perspective, it's useful to quickly take a deep dive into the past 10+ years of AWS billing file updates:

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.

2012: CAR (Cost Allocation Report)
: Back in 2012, AWS users wanted to start answering questions like, "Okay, 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 spend 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)
: Codenamed "Origami" during development, the CUR file was a complete rethinking of the billing schema. For the first time, AWS moved away from comma-separated values into a 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 okay, 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.

Once you know this short history, it's easier to see how it follows the same Crawl, Walk, Run approach. We can summarize the history this way:

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 behavior and rate application, as well as identify where your allocation gaps exist. (DBR-RT)
5. And finally (or at least for now, since FinOps is always evolving), 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)

That evolution also maps to the journey at Apptio Cloudability. 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 platforms 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 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 moves past the Crawl stage.

Hourly data is required to do higher-level FinOps functions like RI planning. RI 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 (our story is still in the Crawl stage), but remember down the road that the fine-grained visibility AWS, GCP, and Azure 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 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 'zombie' instances, we rightsized some things, we bought a few RIs." 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 seems 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 isn't monthly like the data center billing 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 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 in on-premises, where you can set a consistent rate for resources across a time period, cloud rates can vary wildly based on whether a reservation or volume discount was applied by the cloud provider.

Further, amortization of early prepayments of RIs or CUDs 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 charge back. 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 upfront 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. You'd make your teams feel great about their cost reduction, but you'd also give them a false sense of efficiency.

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

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 of a resource used (or the number of seconds used in AWS or GCP clouds). 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. Increase either of those items and your cloud bill goes up. Increase both and it can go way, way up.

This simple formula 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.

Two Levers to Affect Your Bill

With the previous formula, two basic levers are presented to affect cloud spending.

The first lever is to reduce what you use. This is called *avoiding costs*, 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 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 eight years of talking with hundreds of companies who have done it poorly and a (smaller) number who have gotten to the Run stage, we have a firm opinion on this:

> The most successful companies 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. 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 a rightsizing report 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. Give the engineers/application owners the power to make the right decisions for the business.

On the other side of the coin, these same application owners are not typically great at remembering to buy RIs or CUDs. Worse, they sometimes misunderstand the mechanics of how those work. 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.

> 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

Rate reduction offerings like RIs and CUDs can be complex to understand, and cloud service providers are constantly innovating on their offerings (it won't be a surprise if there's a new flavor of them by the time this book is published). Trying to get each

distributed team to spend time learning how these discount programs work 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 RI 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, or other waste. 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 14 on RI strategies, there's more to consider than just resource type when making RI commitments. There's also a conversation about the cost of the capital a company deploys to purchase them that can be nuanced and involves the finance teams.

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 commit to RIs 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.

At the Walk and Run stages, keeping reservation coverage up and waste 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 reservation). The FinOps team lowers the potential for waste while optimizing coverage for all.

One caveat: there may be some teams that, for one reason or another (say, very unique resource requirements), it does not make sense to centralize. We'll cover what to do in that scenario in Chapter 14.

Why You Should Decentralize Usage Reduction

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, generate usage optimization recommendations centrally. After gathering usage from monitoring tools, lean on a FinOps platform to generate alternative recommendations for resources that better match the workloads that appear to be running on them. Then push those recommendations out to the teams on a regular cadence, on a threshold alert basis (for a Walk stage company), or

into developer sprints via JIRA (in a Run stage company). We'll cover the key criteria for those recommendations in deeper detail in Chapter 11.

This model ensures that your resource owners—who have a better understanding of how the resource is used, what depends on it, and whether there are any business reasons for why changing it would impact your running 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, 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. You also should standardize your organization on a consistent cost metric format. That will likely be an unblended and amortized rate that factors in any custom discounts you may have with your cloud provider and the 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 almost all cases.
- Detailed analytics of your billing data is possible only with a deep understanding of that billing data. Learn the raw data to truly become a FinOps expert.
- 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 RI and CUD commitments 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 recommendations by the central team, as these decentralized application owners are the ones best positioned to make infrastructure change decisions.

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, and finally what insights the cloud bill provides. The next section gets into the fun part: implementing the FinOps lifecycle in an organization.

PART II
Inform Phase

Now for the fun stuff: how to begin implementing FinOps in your organization. In this part we cover the *inform* phase of the FinOps lifecycle, walking you through the importance of reporting, surfacing cloud costs, monitoring the performance of your optimizations, and other ways of measuring your cloud financial management success.

CHAPTER 6

The FinOps Lifecycle

Back in Chapter 1, we discussed the core principles of FinOps. Principles are great as they help guide actions, but they need a framework in order to be implemented. Here's the FinOps lifecycle framework you're going to use: Principles → Phases → Capabilities.

The Six Principles of FinOps

We touched on the principles of FinOps before, but now we're going to illustrate how they play out in action by looking at their real-world ramifications, each with specific capabilities designed to yield exact results. The principles again are:

- 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 unpack each of these in more detail.

Teams Need to Collaborate

First and foremost, FinOps is about cultural change, in terms of breaking down the silos between teams that historically haven't worked closely together. When this is done right, the finance team uses language and reporting that moves at the speed and granularity of IT, 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 ensure innovation and speed still flourish.

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. In actuality, cloud is a value creator, and the more you use it, the more cost it will incur. The role of FinOps is to help maximize the value created by the 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.

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.

FinOps Reports Should Be Accessible and Timely

In the world of per-second compute resources and automated deployments, monthly or quarterly reporting 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 cloud resources. 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 are based on fully loaded and properly allocated costs. These are the true costs for operating in the cloud. These costs should be amortized, to reflect the actual discounted rates a company is paying for cloud resources; 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 bad data and hamstring value creation.

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 cheerleading. 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 using less, and centralize paying 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.

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 backward-looking, "How can you ensure you stay within your budget given what you're already using?" perspective. Instead of basing capacity purchases on possible future demand, base your rightsizing, volume discounts, and RI/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.

The FinOps Lifecycle

Now that we've defined the core principles, let's explore how they're implemented across three distinct phases: inform, optimize, and operate (see Figure 6-1). 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 processes that make the goals of IT, finance, and business achievable. Automation can be deployed to enable these processes to be performed in a reliable and repeatable manner.

That's right. The lifecycle is inherently a loop. The most successful companies take a Crawl, Walk, Run approach and get a little better each time they go through it. Let's review each phase and the actions you'll take as you pass through it.

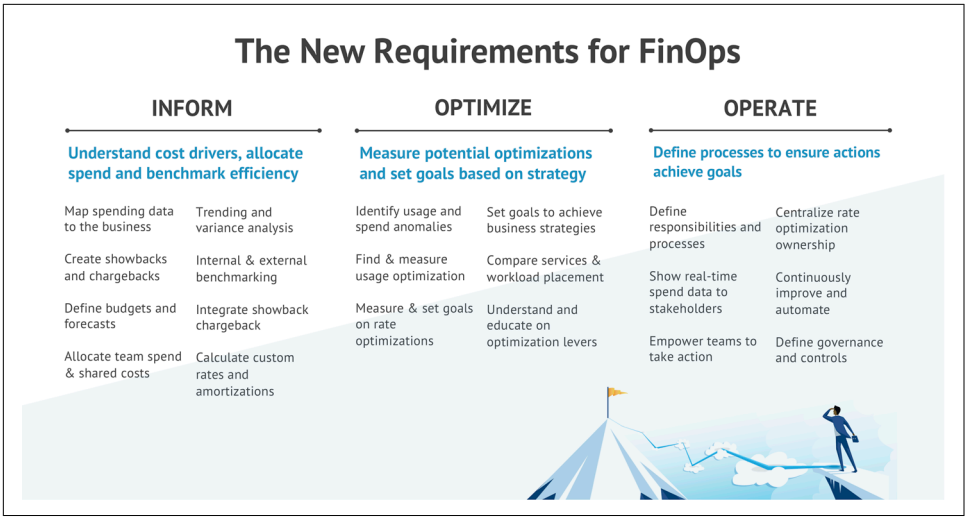


Figure 6-1. FinOps phases and their capabilities

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 behaviors. During the inform phase, you get visibility into IT spend, drill down into granular cost allocation, and create shared accountability. Teams learn what they're spending down to the penny, and why, by using various benchmarks and analyses. For the first time, individuals can see the impact of their actions on the bill.

Some of the activities you'll do 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 showbacks and chargebacks
: As organizations adopt the FinOps model of pushing spend accountability to the edges of the organization, they are finding that chargeback and showback models are becoming increasingly important to driving ownership of spending and recovering costs.

Define budgets and forecasts
: Using the data that's available to them, a FinOps team should be able to generate forecasts of cloud usage for different projects and propose budgets for different

projects. These budgets and forecasts should consider all aspects of a cloud architecture, including containers. Managing teams to budgets (see Chapter 7) 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.

Set tag strategy and compliance
: Tagging strategy, which we'll cover 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, and tag sprawl quickly makes the 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 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/CUDs are applied, and that amortized prepayments from RIs/CUDs are applied. This ensures teams are tracking to the right spend numbers and aren't surprised if their finance team's bill doesn't match their daily spend reporting.

Integrate showback/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).

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.

Optimize

The optimize phase institutes 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:

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.

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 that engineering teams will use during the operate phase, following a predefined process to rightsize resources.

Evaluate centralized Reserved Instances or Committed Use Discounts
: As a cost-reduction measure, the FinOps team can evaluate metrics on existing AWS/Azure RIs or GCP CUDs to make sure the ones they have are effective and

then look for opportunities to buy more. 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.

Compare prices and workload placement
: Workload placement is another cost reduction measure. Once the FinOps team understands engineering's infrastructure requirements, they can look at multiple cloud vendors and compare pricing options.

Operate

Where the optimize phase sets the goals for improving, the operate phase sets up the processes for taking action. Goals aren't set here, but rather decisions and plans are put into place to address those goals based on the identified business case context. 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 they want to continue operating them as they have been, or whether they can make some changes. Here are 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 cloud and level up their reporting efficiency.

Rightsize instances and services
: 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 hard drive rather than SSD-based storage for some projects. 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*. 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 rightsizing recommendations into sprint planning tools such as JIRA.

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. In essence, 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.

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 pivoting around 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. As Figure 6-2 shows, culture and governance are the central core that holds everything together and ensures FinOps success.

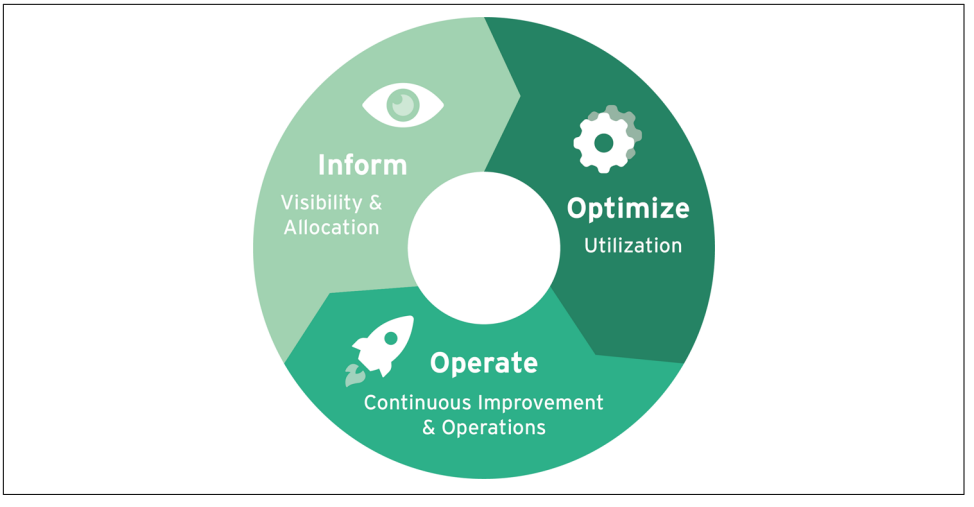


Figure 6-2. The FinOps lifecycle revolves around culture

Whatever you do, don't try to boil the ocean. Follow the Crawl, Walk, Run approach. Years ago, a major retailer tried to go from 0% to 80% RI coverage in a single purchase. The company studied its infrastructure, consulted its engineering teams, checked its OSes, 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 OS 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, mastery takes time and learning from those who have gone before you.

Why to Start at the Beginning

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 6-3).

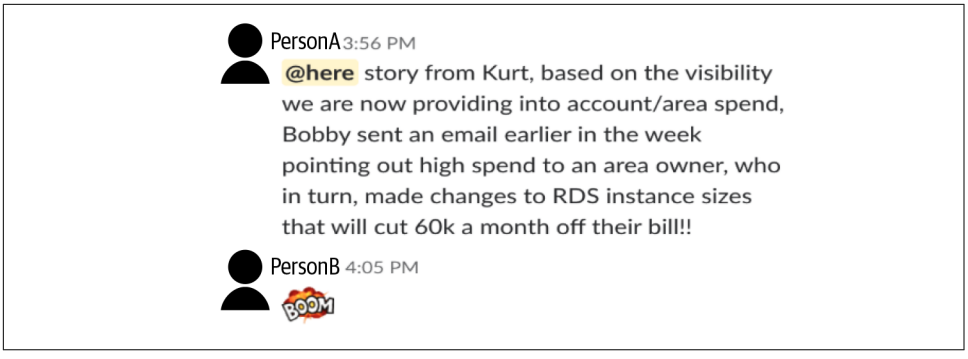


Figure 6-3. A real conversation about the results of cost visibility

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. 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.
- Start with a Crawl approach in each phase—don't try to do everything at once.
- Involve all your cross-functional teams early and often so they can learn to crawl 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 to 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 walk through the first phase of the lifecycle so you can start addressing the questions you need to answer.

CHAPTER 7

Where Are You?

You 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 in the optimize phase.

As we've stated, FinOps isn't a linear process. But the visibility of the first phase is essential for equipping 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. In other words, you'll always circle back around to inform, and check in again.

You need to generate data to get an idea of where you stand. Then you'll be poised to take actions that will be beneficial to the business.

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, IT, 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 8. Remember, you're not trying to boil the ocean. You want to identify questions that must 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: A Novel About IT, DevOps, and Helping Your Business Win*

Even the most experienced FinOps practitioners—those who have built multiple mature practices—follow the Crawl, Walk, Run process when setting up a practice. Your initial goal is to get some easy wins, harvest the low-hanging fruit, and gain credibility quickly. You should think DevOps, not Waterfall. Think Gene Kim, not Winston W. Royce.

Seek First to Understand

The first question you should try to answer is, "What's the total 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.

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 it about seeing trends or about being able to accurately charge back to the penny? Early on it will be about trends; later it will be about getting very granular.

- How will you account for changes in cost centers? Early on, you may have a spreadsheet, 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 shifting between 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 operate 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:

- *Crawl:* Which teams are driving the costs? *Walk:* Are they being efficient? *Run:* Can you link their costs to unit metrics?
- *Crawl:* Do you have budgets in place for each team? *Walk:* Are you managing teams to those budgets? *Run:* Are you able to do activity-based costing?
- What's the state of your tagging strategy? *Crawl:* Look at what tags are in place and what they are reporting. *Walk:* Look at coverage gaps. *Run:* Consider how to allocate untaggable or shared costs.
- How will you keep your allocation constructs in sync and up to date? *Crawl:* A spreadsheet. *Walk:* Cadence-based syncing with your CMDB. *Run:* API integration between your FinOps platform and your CMDB.
- What is the RI or CUD strategy? *Crawl:* What RIs are there? *Walk:* How well are they being used? How could they be exchanged or modified? *Run:* Which new ones should you buy? And then…*rinse and repeat*.

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 the Crawl of cost visibility. Then in the next cycle, you set basic budgets. And the cycle after that, you're managing to 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 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 (*http://FinOps.org* has more detail)
- Bridge-building with colleagues in finance or IT, depending on which side of the house the FinOps team is on
- Evangelizing FinOps work internally via events like FinOps Day, to help share the concepts, impact, and early wins (resources available at *http://FinOps.org*)
- Aligning with engineering teams (and helping to take work off their plates) by managing as much of the rate optimization as possible

Transparency and the Feedback Loop

In Chapter 1, we explored the Prius Effect and the impact of real-time visibility on our 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 critical real-time data to drive future decision making and build accountability.

Recently, we were asked, "Is 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 cloud, which is due less to the computers themselves and more to the human behavior that's driving cloud innovation. During the Crawl stage, reports are viewed daily. They're viewed during the Walk stage on an agreed-upon cadence (or based on anomalies), and in the Run stage they're viewed after an alert showing that a metric you've set has

crossed the threshold. We'll dig into what that Run stage scenario looks like in Chapter 16.

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

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.

Potentially 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 Run stage company. A Walk stage company 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.

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 recent FinOps Foundation call, Intuit's Dieter Matzion shared his approach to scorecarding. The key items were:

- EC2 efficiency via a rightsizing score
- Reserved Instance 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.

Forecast and Budgeting

If you're trending off of your forecasts, you must ask yourself: does something need to be changed to bring you back to 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 7-1). 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.

The general consensus in FinOps Foundation meetings is that cloud team forecasts should be done on a 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.

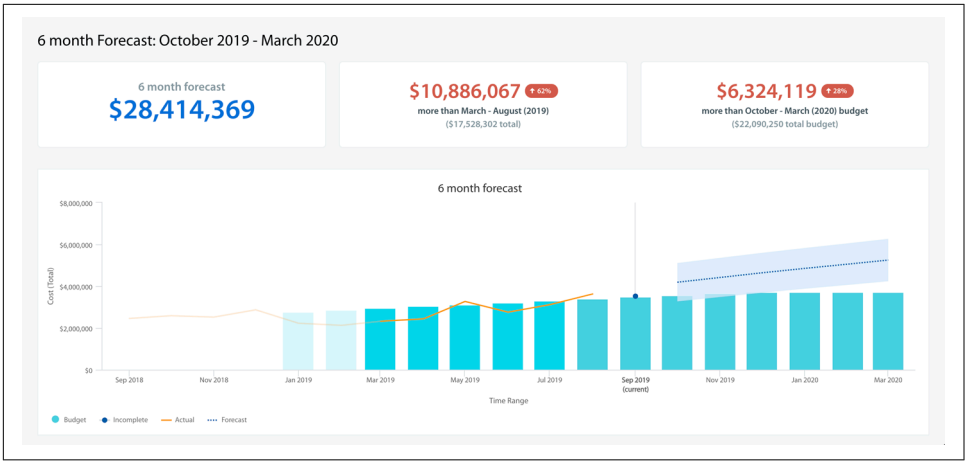


Figure 7-1. A single graph showing the six-month forecast, projected budgets, and actual budgets/use for the previous six months

But evaluating forecasts on a quarterly or monthly basis isn't frequent enough. Skilled FinOps practitioners constantly (daily) evaluate variances based on live data to see which teams are deviating from quarterly forecasts and then lean in with them accordingly. This lets them get in front of variances before they grow larger. Changes in the forecast can then be reported up the management chain the day a decision is made to spend more than planned. Conversations can take place between management and the business to determine the value, if any, that the increased spend is generating.

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 should provide a vehicle for identifying when spending deviates from plan—which, remember, enables conversations.

Forecasting should be based on machine learning. Matzion shared with the FinOps Foundation that manual forecasts were often 20–70% off from actuals, even after humans had agonized for several weeks to analyze and predict the data. Those results just aren't worth the effort.

Machine learning is the key to accurately predicting spending. But a single model or algorithm isn't enough. Some machine learning algorithms work better with a longer history of cost data, while others are able to work with limited data points. It's likely you will have accounts with a long history of data, but when you introduce new workloads or brand-new accounts, some machine learning 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.

When forecasting, you must take into account some important considerations. What is the period of time you're basing your forecast on (spending patterns may be different from six months ago)? What is the cost metric you're using? Will it be amortized to include RI purchases? Is it adjusted for custom rates? Alignment between teams is key, so you have to be sure to agree on the cost metric early on.

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. Under 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—that's 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 see 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. And the cost-optimization conversation has recruited them as allies and given them visibility into how they can help the business. They begin to understand that FinOps is not about cutting resources but about enabling speed and innovation.

For many tech unicorns, budgeting is a four-letter word. It's a necessary evil that distracts from trying to hoover 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. And 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 the 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.

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. While 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 runaway 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 what you spent in the last period 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.

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. 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 19.

What Great Looks Like: Crawl, Walk, Run

There is no shortcut to becoming a high performer. Building organizational muscle will take years to develop. Of course, you can do things to help speed up the process, but ultimately you need to go through a series of learning processes. This education and cultural development within an organization takes time. 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 7-1 shows various levels of proficiency across key capabilities in the Crawl, Walk, Run lifecycle. Quantitative data was taken from the more than $9 billion of public cloud spend in Apptio Cloudability's dataset, while qualitative data was taken from a survey of hundreds of cloud consumers by the 451 Group.[1]

Table 7-1. Capabilities 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

The 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 the granular visibility and allocation of cloud spend, they know what needs to be done to optimize their operational metrics.

1 451 Research, *Cost Management in the Cloud Age: Enterprise Readiness Threatens Innovation* (New York: 451 Group, 2019), *https://oreil.ly/71tao*.

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 perform optimizations and operational improvement in the next phase.

To summarize:

- Build context around your financial data in order to answer questions.
- Use data to monitor spend and plan for optimizations and efficiency.
- Forecasts and budgets should be used for early detection of issues causing your cloud spend to deviate from what is expected.
- Budgets are a necessity that doesn't have to be painful when tied to unit economics performance metrics.
- The Crawl, Walk, Run model is important. With each development of maturity, you circle back, which enables you to answer even more complex questions about your cloud.

We've identified the questions, but without a solution to divide cloud spend into cost groups, only the overall questions are answerable. You need a method of identifying which costs belong to which group in order to perform team-by-team budgets and forecasts. Then you can benchmark teams and compare them to each other. This is the subject of the next chapter, in which we introduce the concept of cost allocation.

CHAPTER 8

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 know 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.

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. 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 of 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.
>
> —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 7. 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 that a cloud noob 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 cloud. Even a solid linked account strategy or a decent tagging strategy isn't enough to solve the issue of allocation. Partially, that's because allocation can be a moving target. Consider untagged or untaggable resources, shared costs, and 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 groomed it might seem.

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 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. Context is everything. So as you build your strategy, you'll be keeping all of the different needs and points of view in mind so that you can create allocation constructs that everyone finds meaningful.

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.

Chargeback Versus Showback

There are ongoing conversations in the FinOps Foundation community about when to do chargeback or showback. Sometimes the choice is a function of an enterprise's maturity. The less mature organizations do showback, and the more mature ones do chargeback. 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 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[1] by Barry Whittle on Emerge, Apptio's thought leadership property and the home of Technology Business Management:

> Showback leaves a bill on the table for analysis; chargeback leaves the same bill, but requires payment.
>
> —Barry Whittle, Marketing Communications Writer at Apptio

In practice, the choice between the two is more often informed by the structure 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&Ls. 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.

A Combination of Models Fit for Purpose

During a recent FinOps Foundation call, Ally Anderson, a business operations manager (and five-year FinOps veteran), 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.

Anderson confirmed that allocating spend among teams is complex and ongoing. She maps each team's dozens of disparate linked 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 or VP.

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.

Anderson shared how her ever-maturing approaches to ensuring allocation happen, and they mirror what we've seen at other enterprises:

Account naming
: Each time there is a request for a new account, it must be named for the product that it will be supporting. And if it's environment-specific, the environment must

1 Barry Whittle, "Showback: The Essential Guide to Drive IT Spend Accountability," May 14, 2019, *https://oreil.ly/fu73M*.

be represented in the name. Anderson, as a FinOps practitioner, is the central checkpoint to ensure this process is followed. All requests for accounts must go through her. In AWS, this would take the form of linked accounts. In GCP, this would be in the form of projects or folders.

Tagging
: Because not all of her company's legacy accounts were opened under the new account naming convention, Ally also makes heavy use of tagging, especially in the legacy multitenant accounts. Their primary tag key is called "Department," and it helps them carve out spending in those accounts. The legacy multitenant accounts require that tag to 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 sizeable chunk of untagged spend that is reported out to management.

Shared corporate costs across accounts
: Anderson also spoke on the common problem of how to allocate central costs in the cloud world. In her management reporting, she splits and spreads out costs like volume discounts, enterprise support, and Reserved Instance prepayment amortization.

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 Anderson 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.

Following a typical maturity curve in FinOps, Anderson 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, Anderson has matured out of the Crawl phase, 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. Anderson shows each team their actual spend against forecasted spend, including the variance between the two and a comparison to what was budgeted. She does

this on both a monthly and quarterly basis, as well as providing regular, more granular reporting into cost drivers and anomalies.

The CTO has a monthly operating review with each of the engineering VPs to show their spend. If there aren't any glaring variances, these meetings tend to be quite brief. Larger variances bring a more focused approach. The real-time FinOps processes Anderson has in place give her VPs an as-early-as-possible heads-up that they will overshoot their budget.

When a variance comes to light, Anderson can quickly highlight the issues and the cost drivers that caused the variance, and then the VP can work with the team to implement a fix without waiting for the monthly operating review. It's 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 on a fully amortized, custom rate, adjusted and unblended to show the actual cost a team incurred in the period. The most mature organizations also split up shared costs such as support charges.

Centralize RIs 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 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.

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 8-1 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 8-1. 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 8-2, 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 8-2. 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.

Amortization: It's Accrual World

Recall the matching principle we 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 RIs, where a portion of the initial expense will need to carry forward to when the RIs 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 upfront 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 8-1, 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 becomes clearer once costs are amortized.

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.

  $200 / (365 \times 24) + (0.05) = \$0.073$

- The savings works out to be the on-demand hourly rate minus the amortized rate.

  $0.10 - 0.073 = \$0.027$ per hour (27% savings)

A company might have hundreds, if not thousands, of RIs at any given point in time. This makes it quite challenging to track all active reservations. Each reservation would typically have a unique schedule, with a certain start and end date. In addition, 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.

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.

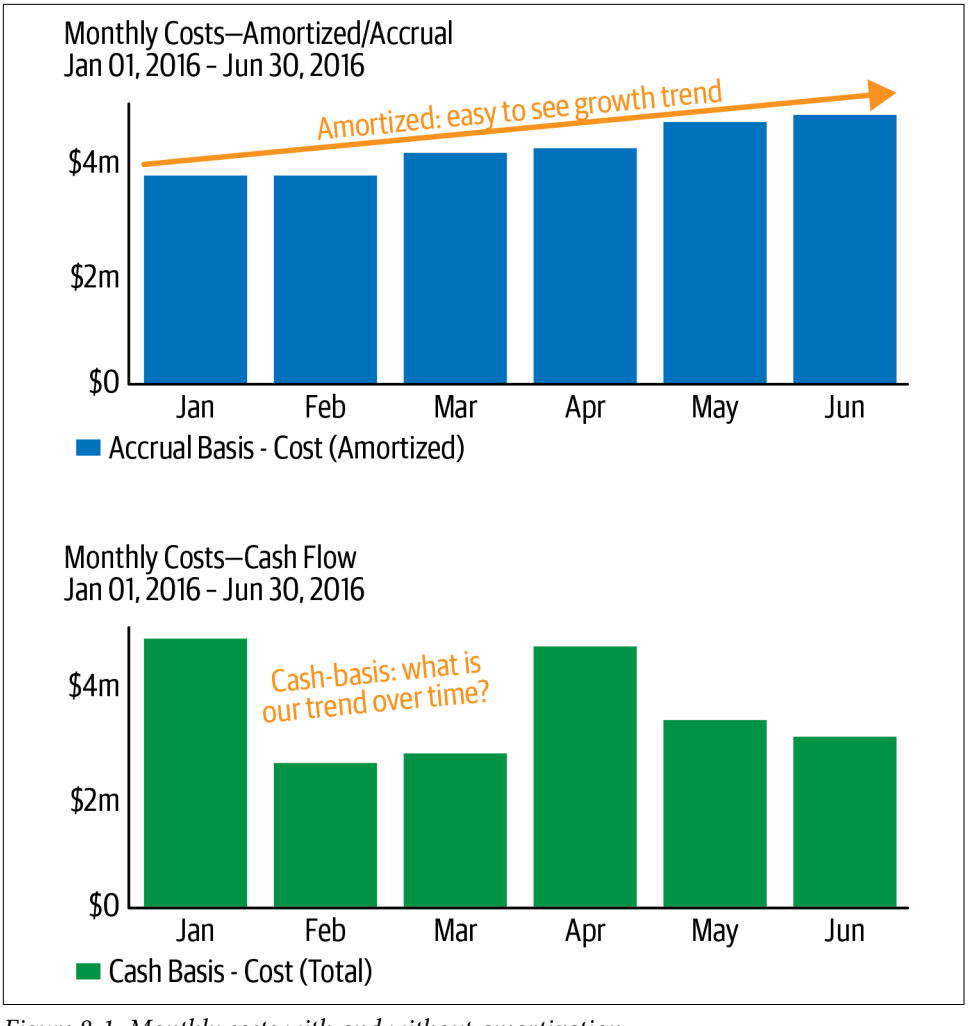


Figure 8-1. Monthly costs with and without amortization

Creating Goodwill and Auditability with Accounting

The additional benefits of 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 Gajewksi, 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.

Going Beyond Cloud with the TBM Taxonomy

You gain benefits from mapping spend against your own financial structures like cost centers and object codes. But you should also consider tying it back to a consistent taxonomy that you can use to benchmark and integrate your spend data in views that are meaningful to your business partners. Finance teams don't always need details on how specific cloud services were consumed by the business, but they often need to allocate the spend against the right cost pools and IT towers to provide a consistent view across disparate teams, and for benchmarking against industry peers. CIOs often want to hold internal consumers accountable for their cloud spending, which is difficult to do without proper mapping. And infrastructure leaders need to compare on-premises technologies with public cloud services in an apples-to-apples way, a process that is frustrating if you don't have commonly understood terms and definitions.

Many scaled enterprises (as much as 50% of the Fortune 100[2]) use the Technology Business Management (TBM) taxonomy to accomplish this across their broader IT spend. There are many parallels between TBM and FinOps, but they also give us different perspectives on data.

The TBM taxonomy[3] classifies and organizes IT costs from different sources in a broad, standardized hierarchy. It provides a common set of terms to describe cost drivers and consumption—and ties all IT spending (regardless of sourcing model) back to business value. More than just a way to categorize spending, the consistent structure of the TBM taxonomy enables both internal and external comparisons of technology spending. It helps align data center spend to public/private cloud spend by normalizing the way they are reported. The consistency also makes benchmark comparisons easier between these typically divorced technologies, and also with other companies in similar industries.

2 Sarah Vreugdenhil, "Apptio Opens First Center of Excellence in India to Recruit Top Talent and Help Companies Fuel Digital Transformation," August 22, 2019, *https://oreil.ly/1Kt5R*.

3 TBM Council, "The TBM Taxonomy," November 22, 2018, *https://oreil.ly/5pQok*.

The TBM taxonomy (see Figure 8-2) provides a standardized hierarchy for reporting costs and related business metrics.

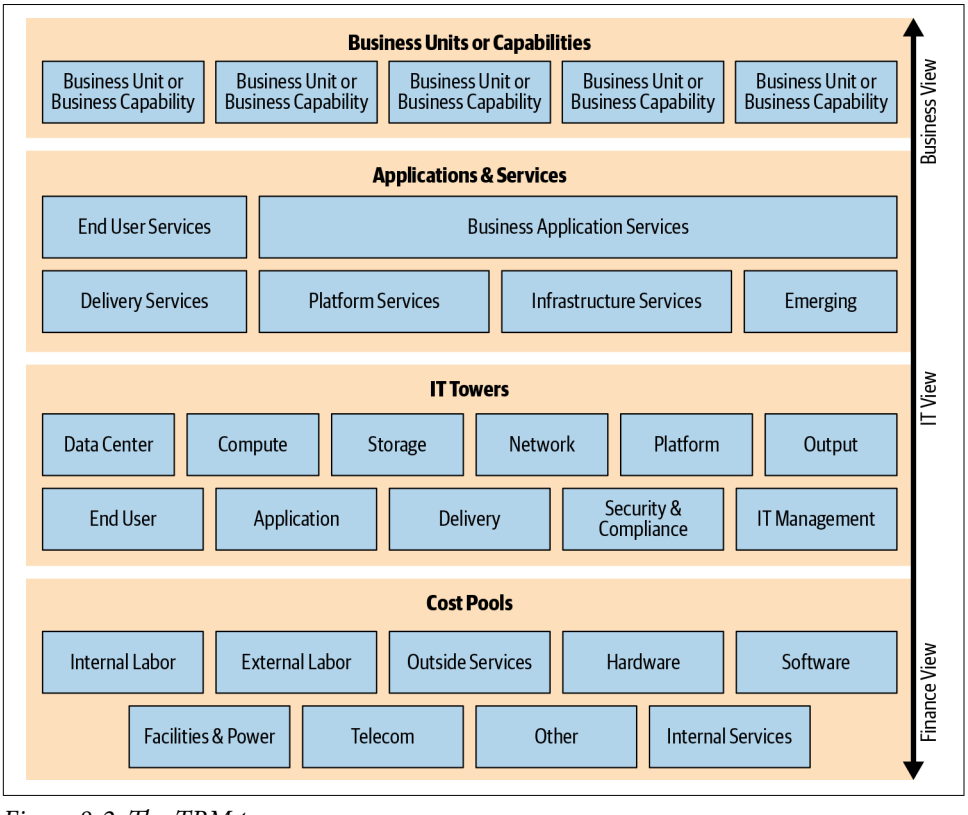


Figure 8-2. The TBM taxonomy

The four layers of the taxonomy (cost pools, IT towers, applications and services, and business units or capabilities) support three different views into the data for different parts of the business. As we discussed earlier, businesses historically have wanted to see spending data through different lenses and contexts.

For finance, the model incorporates general ledger (GL) data and additional cost sources from across the organization. It also helps differentiate between CapEx and OpEx spending—as well as fixed versus variable spending—and pools all of the IT spending into the nine standard cost pools (Hardware, Software, Internal Labor, External Labor, etc.) represented in Figure 8-3. This helps finance teams with cost allocations and makes it easier to compare internal teams, projects, and other cost drivers.

For IT, the model maps spending from the cost pools into common functions (e.g., server computing, storage, application development, application support, IT management), which are called *IT towers*. Because these towers are highly consistent across enterprises, they enable IT to assess and benchmark their efficiency between all of their IT spend, whether it be cloud or internal. It also enables them to monitor unit costs over time, set targets for improvement, and manage their efficiency largely from a supply-side perspective.

For the business, the model maps spending from the IT towers into standardized products and/or services that in turn support distinct business capabilities and/or are consumed by specific business units. For example, IT might provide infrastructure-based services, such as application hosting, data warehouse, or networked storage. In addition, there will likely be application-based services, like customer relationship management, financial management, or product engineering. The TBM taxonomy gives you a uniform way to map into these types of products and services. And at this layer of the TBM taxonomy, the terms make perfect sense to the business consumers, be they business application owners or business unit executives.

Most important, the taxonomy provides a set of predefined metrics and key performance indicators (KPIs) in a standard model, so that the business can have conversations about how cloud spend (and, in fact, all types of IT spend) ties back to business value. Further, it can help to accelerate benchmarking across different teams and types of spend.

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 8-3 shows a sample journey into 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.

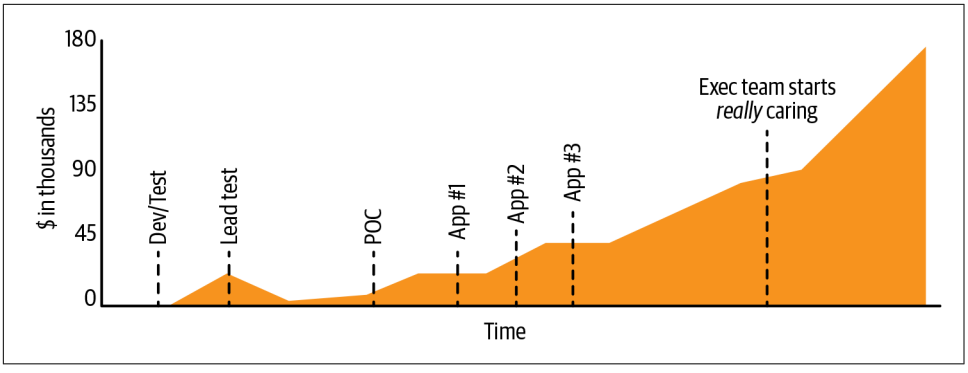


Figure 8-3. Spend growth with significant events

Ally Anderson 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, Anderson 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 repeating the Crawl, Walk, Run process of FinOps.

Spend panic is going to happen. If you have an automated system in place to see your spend patterns, and an approximate date when you know spend will increase too much, CCoEs can be proactive and prepare for the inevitable executive scrutiny that will follow. Without a CCM or defensible cost allocations, you are sleepwalking into an avoidable conflict with business leaders. You can protect yourself, and your sanity, with a scalable, easily updated allocation strategy.

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 7.

To summarize:

- Chargeback and showback each drive accountability.
- Chargeback is the most advanced model, but it's not right for all companies. It's often best to start with showback and then gain organizational support for moving to chargeback.
- You must be sure to factor in shared costs to the model, such as support, amortization, and unallocated expenses.
- A showback or chargeback model can also help with better tagging and auditability.
- 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 9

Tags, Labels, and Accounts, Oh My!

In the previous chapter, we discussed why cost allocation is essential for cloud financial management. Whether it's done via tags, linked accounts, folders, or labels, cost allocation is essential for cloud financial management. Without a consistent allocation strategy, you have a big pool of costs with no way to split them up or to identify what costs belong to which team.

Although tags, labels, and accounts provide many benefits, one major advantage is that they add business dimensions and context into your billing data for cost allocation. Throughout this chapter, we'll cover the two major methods of allocating costs: hierarchical account separation and tagging. We'll cover how they each contribute to cost allocation, and we'll describe some successful strategies we have seen.

Each cloud provider uses different terms for similar concepts; 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 GCP "labels." When we say "accounts," we're also referring to GCP "projects and folders" and Azure "subscriptions."

Cost Allocation Using Tag- and Hierarchy-Based Approaches

As we discussed earlier, a cloud service provider will supply a detailed bill that breaks out spending in fine detail. Depending upon the size of an environment, this might run to millions or even billions of lines. For every resource you have consumed during the month, there will be details about the cloud resource involved, where it was run, how much you used, and the rate you were charged.

What's missing from all this detail is the business logic about the resource: 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 questions 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.

Every team 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 what business activities are driving costs.

The key to winning at FinOps is getting this allocation strategy up and running, and doing it early and consistently. To do so, you need to get 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 (or labels) that can be applied to a resource, as well as hierarchy-based accounts/subscriptions/projects/folders/resource groups in which resources are run.

Some cloud service providers even provide solutions that allow you to group and build deeper hierarchy-based allocations. GCP folders and Azure resource groups can be used in addition, or even as an alternative, to tags on individual resources. AWS provides the ability to tag accounts, and GCP allows tags on projects. This capability may be available from other providers soon.

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 accounts/subscriptions/folders. But you've got to use one of these approaches, and ideally—especially when your environment becomes complex—you use both. And you apply them with a consistent strategy. Generally, if you're going to pick 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. Time and time again, we've seen a tag-only approach lead to missing coverage, resulting in a lot of unallocated spending. We found that not only are engineers notoriously bad at ensuring all of their resources are tagged, but also the cloud service providers generally have a 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 will 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 mutating the billing data and supplementing it with extra detail using a third-party analytics tool such as Apptio Cloudability 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 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 in order to understand how your allocation strategy can be affected by their various offerings (see Table 9-1).

Table 9-1. Comparison of cloud vendor allocation options

	AWS	Google Cloud Platform	Microsoft Azure
Hierarchy	Linked accounts	Folders and projects	Subscriptions
Key/value pairs	Tags	Labels	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
Tags can be applied to	Most services' resources (constantly changes; see AWS documentation for all current details), accounts (via AWS Organizations)	Most services' resources and projects	Most services' resources, Azure resource groups

From Table 9-1, 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 GCP calls them *labels*), but there are differences in the limits each provider applies to these tags.

In addition, cloud providers are adding 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 9-1, 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.

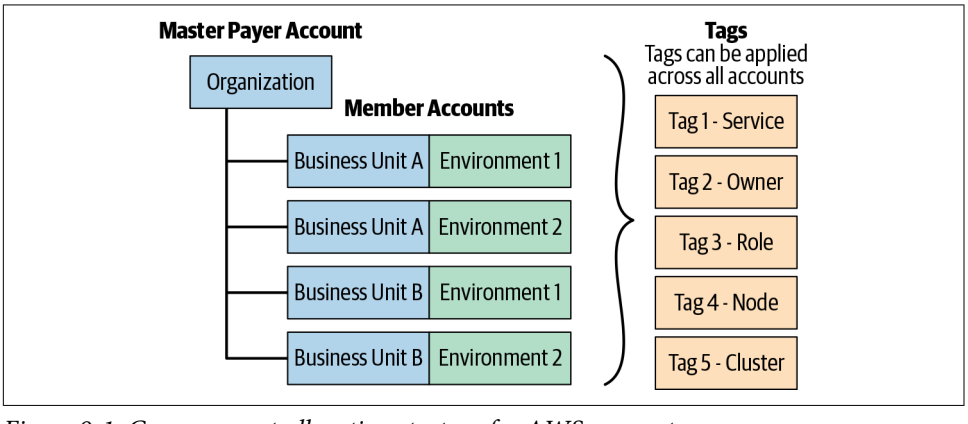


Figure 9-1. Common cost allocation strategy for AWS accounts

In other 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 challenge tag-first companies often run into comes when they apply a broad tag policy in order 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 Projects Using Folders in Google Cloud Platform

It's worth noting that GCP has an added layer of allocation for projects called *folders*.

According to Google's documentation:[1]

> 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 GCP 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 the diagram below [Figure 9-2], 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.

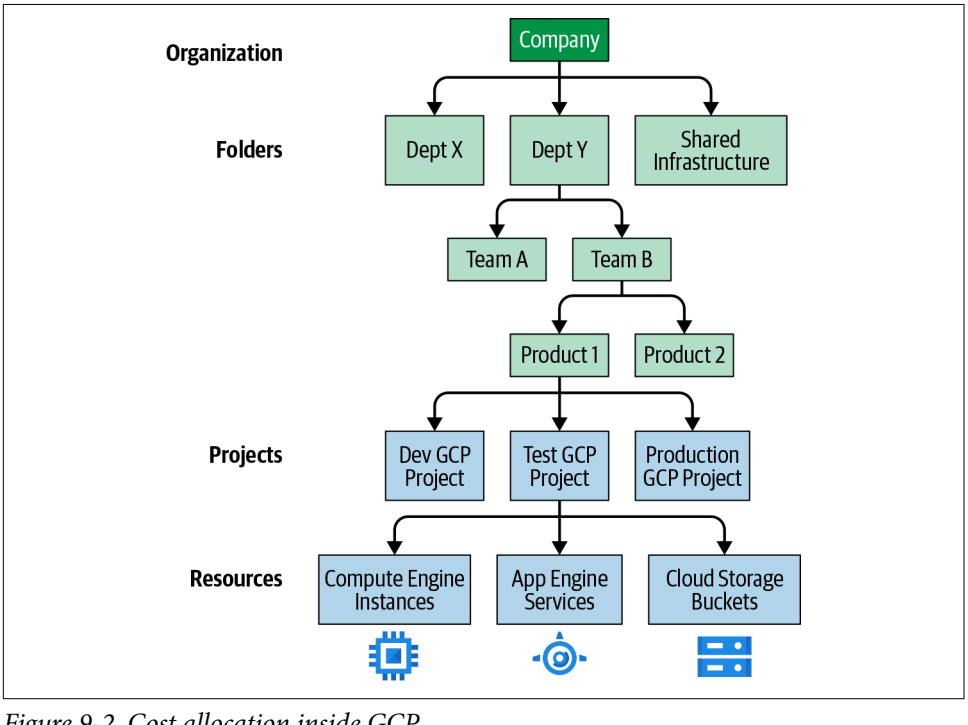


Figure 9-2. Cost allocation inside GCP

1 "Creating and Managing Folders," Google Cloud, last modified September 11, 2019, *https://oreil.ly/7jbAo*.

It's quite common for a company to create folders that in turn contain additional folders or projects, as shown in Figure 9-2. 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 9-3 shows some cloud resources tagged with an *Environment* and *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 (given syntactical limitations such as character set and length). However, having a clear plan for tag keys and their appropriate/approved values will ensure that you do not end up with inconsistently applied tags—or worse, with no tags at all.

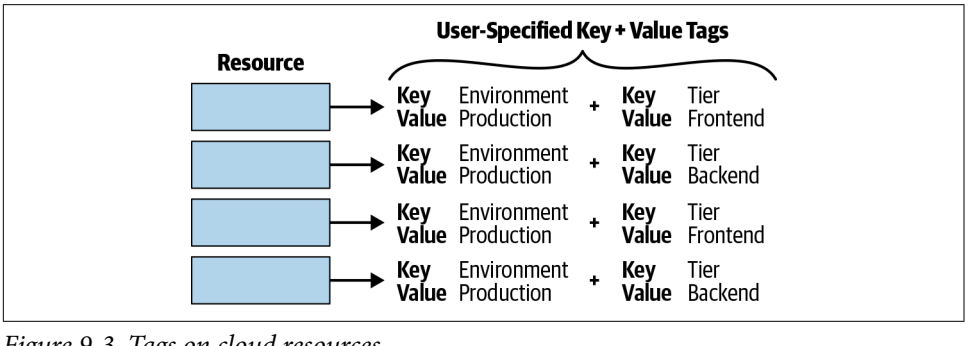


Figure 9-3. Tags on cloud resources

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 allows 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 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 for billing until a later date leads to gaps in your tagging allocations. Then when you need to analyze billing data for the cost allocation and cost optimization strategies we'll discuss later in this book, you find that billing data is of no help to you.

Tagging is a vital early step that helps ensure that by the time you get to building out further FinOps strategies, you'll have a rich data set to work with. That will put you on the road to saving costs a lot sooner.

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 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. However, we've seen over and over that companies

with management buy-in for the new tagging standard have better adoption of tags across the organization. Having management push out the message that tagging is essential not just to the FinOps team but to the business at large will 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. Having a good tagging policy in an organization will continue to pay dividends.

Getting your tagging strategy right the first time is essential. We can say from firsthand experience that going back to a team and asking them to go update all those tags they just finished rolling out will result in major stink eye at minimum.

The more thought-out a tagging standard is at the start, the less likely a future revision will be required. As a business scales out on the cloud, it will be generating thousands, if not millions, of resources, 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. Requiring an excessive amount of tags will only cause resistance to the standard, which will lead to noncompliance. You should always keep in mind the questions you need to answer, and how much of the data you need is being provided by 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. This will allow for correct cost allocation within your billing data.
- 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.

Some optional tags you may consider defining include:

- A *tier* tag to identify the part of the service layer the resource belongs too (e.g., frontend, web, backend).
- A *data classification* tag to help identify the type of data contained on the resource.

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 vendors offer ways to fix inconsistent tags after the fact, such as Apptio Cloudability's Business Mappings feature.

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 RI fees or CloudWatch fees, are one reason we recommend using some form of hierarchy grouping in addition to tags. Then you can allocate as much 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.

Most people understand that if one team uses a tag value of "prod" and another uses "production," these tags would be grouped differently. It's also important to realize that tags are case-sensitive, so the value "Enterprise" is different 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, and 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 services that are 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.

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. She works closely with the technical contacts in each team, and says, "If there's a large amount coming in as 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. She also gets reinforcement from management. 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 the untagged costs.

Darek Gajewski from Ancestry, also a FinOps Foundation member, 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.

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, and that will allow you to move on to 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 near-real-time business decisions that will optimize an organization's cloud. We also look at the cloud service provider's offerings that can help to reduce cloud costs.

CHAPTER 10

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 innovation. There are often many business benefits that can be gained from your cloud spend. 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?

For example, 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.

Tracking your spend growth as you accelerate into the cloud helps ensure that you're within the acceptable bounds of spend. Thus, you should think of the optimize phase as being about measuring spend versus speed.

While teams focus on project delivery, it's essential that the FinOps practitioner is simultaneously working on a plan to review potential cost optimizations. This plan makes the process easier when you finally change gears and slow down to optimize your costs.

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 10-1, is a model of the constraints of project management: speed, cost, and quality. 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.

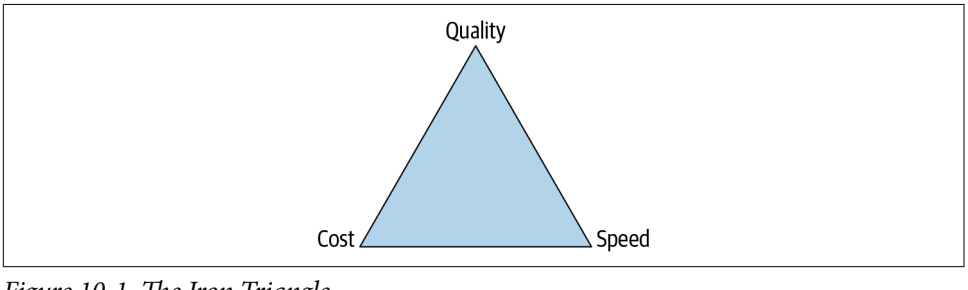


Figure 10-1. The Iron Triangle

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 in order to support their business goals.

Good measures the quality of your services. Giving more resources given to a service running in the cloud often results 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 definitely 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 in order 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, Director of Cloud Optimization 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

At the writing of this book, Daly's FinOps team at Nationwide was relatively new. He offers 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 uses OKRs to build credibility internally. He does this 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: Sustainability

All too often, new FinOps teams approach their work in unsustainable 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 coming in developing policies that make life harder for the engineers. Actually getting people to work together to build better-focused OKRs will drive better results for everyone.
>
> —Joe Daly, Nationwide

Business leaders need to decide how much they are willing to spend, and when they should forgo savings (and potentially even 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 that threshold and then struggling to respond to the high costs, setting early expectations and tracking cloud spend as projects progress will always help avoid bill shock. Engineers are then given freedom within 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.

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

Here are some sample goals shared by a FinOps Foundation member on a recent call:

> 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 commit to the provider and (2) our budget progress within each budget tranche. Finally, we track a *total opportunities to save* number 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 this book.

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 10-2, 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.
- The 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.

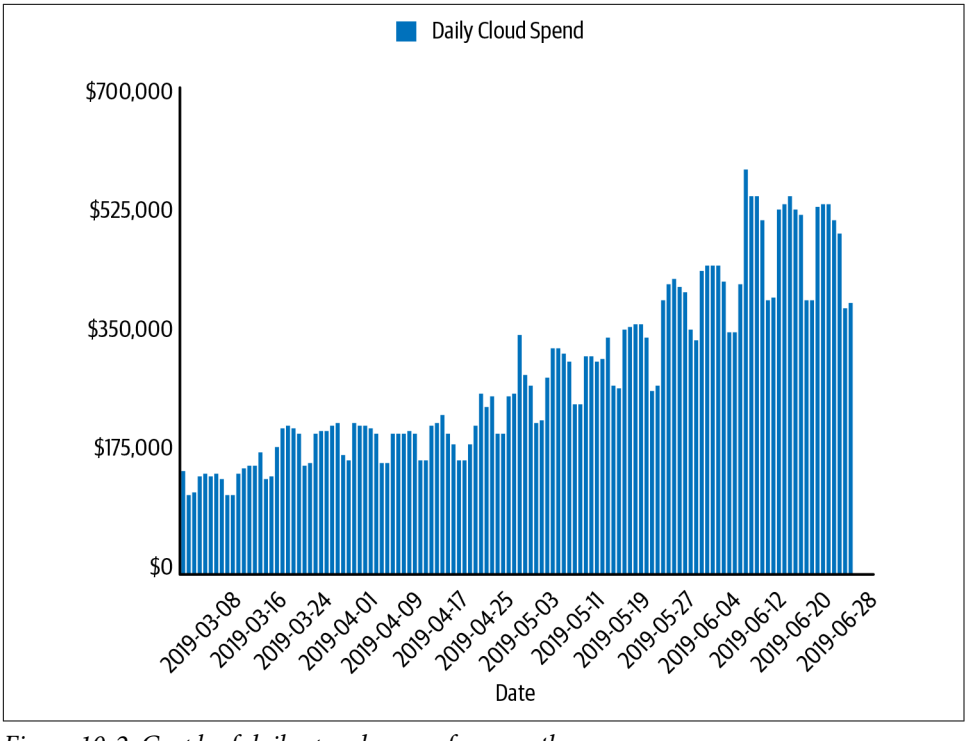


Figure 10-2. Graph of daily spend over a few months

However, if you add a target line, as shown in Figure 10-3, 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. 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.

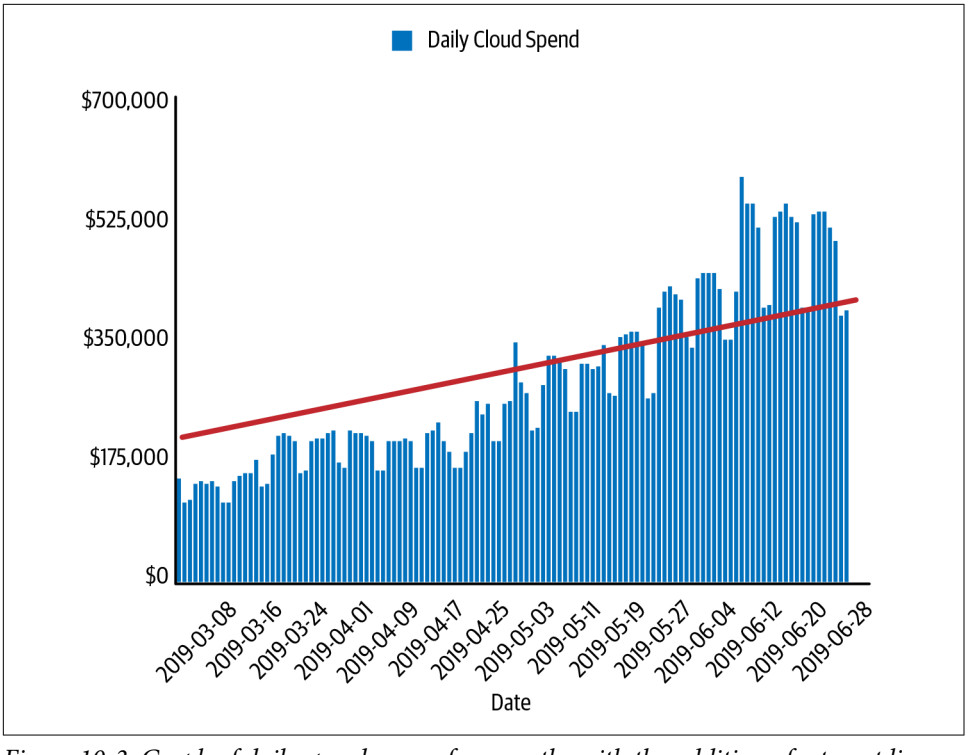


Figure 10-3. Graph of daily spend over a few months with the addition of a target line

Detecting Anomalies

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 decide 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 in order 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 19.)

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.

Reducing Spend to Meet Forecast

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.

Using Less Versus Paying Less

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 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 11 and 12.

Conclusion

The inform phase of the FinOps lifecycle helped 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 11

Using Less: Usage Optimization

In this chapter we discuss one of the toughest parts of the FinOps lifecycle. Usage reduction can be hard and can take time to implement culturally. If quick wins are your goal, you will find that purchasing Reserved Instances/Committed Use Discounts—which we cover in Chapter 13—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 find it.

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 utilizing these instances 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 running at scale in cloud, you can almost guarantee that there'll be some underutilized or forgotten resources. When faced with the apparently limitless resources in cloud, most engineers will tend to go for the larger

resources. Making that choice often keeps 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%. Unlike reservations, you make zero commitments to the cloud providers, so you're free to resize an instance—or to stop using it entirely—whenever you desire. Further savings are possible on correctly sized resources with rate reduction options, like Reserved Instances, which we'll cover in Chapter 13.

Initially, usage reduction tends to be retroactive. Teams go through and clean up their infrastructure to ensure a fully utilized set of resources. The next phase is often rearchitecting applications to use more cloud-native approaches, using cheaper cloud services, and reducing the overall cost of operating services in the cloud.

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.

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. And 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 lie 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.

Find waste by looking for areas with very stable costs and dated resources (old and unchanging is not how the cloud should operate).

Telling teams not to forget about resources doesn't cover all 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 ever taken. 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 the business. 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 cut-off 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 essential to supply multiple recommendations that would fit the existing workload on the resource without clipping the workload performance. Multiple recommendations give teams the opportunity to balance the risk involved in reducing the resource size against the potential savings. A FinOps platform like Apptio Cloudability can do the heavy lifting of providing these recommendations, and engineering teams can investigate further using application performance monitoring tools such as SignalFx.

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.

Common Rightsizing Mistakes

Relying on recommendations that don't account for spikes in utilization

When looking at the existing workload on a resource (like a server instance), it's crucial to ensure that you're looking at peak (or max) values of the metrics. If you use average metrics, 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.

For example, when you compare the two usage graphs in Figure 11-1, 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.

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.

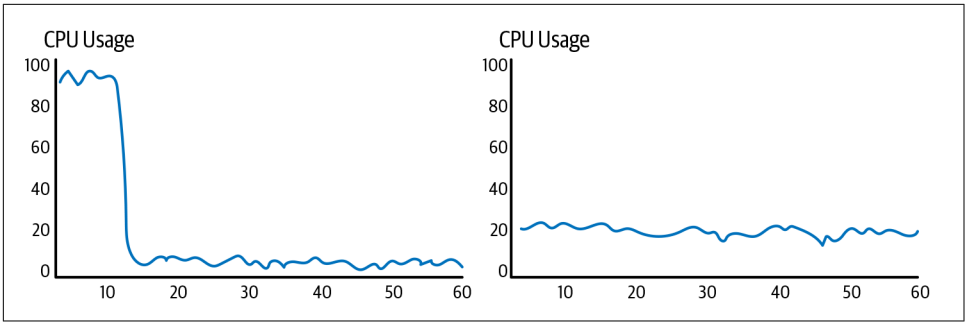


Figure 11-1. Graphs showing two CPU workloads—one has a short 90%+ peak and then remains at <10% for the rest of the hour, while the other is consistently at 20%

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.

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) and storage such as EBS (Elastic Block Store). 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. For example, the memory on a particular r5 instance may be drastically overprovisioned, but the CPU is about right. This is an opportunity to get your shape correct by keeping compute consistent while at the same time reducing your memory. In this scenario, you could save significant money by going from an r5.large to a c5.large.

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 11-2) 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).

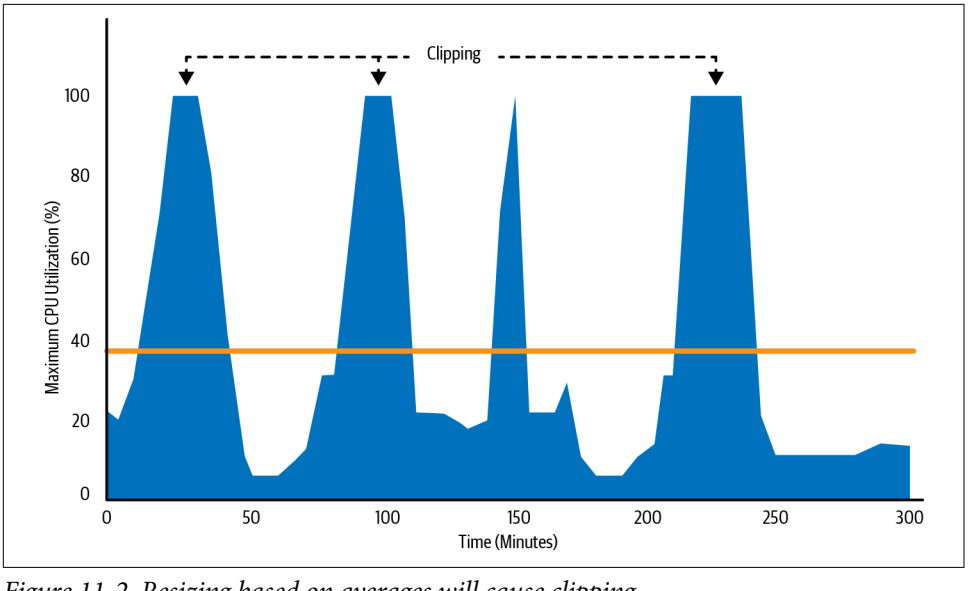


Figure 11-2. Resizing based on averages will cause clipping

Hesitating due to reserved instance uncertainty

A common refrain from teams is, "I don't want to rightsize because it might affect my RI 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 12 and 13.

Going Beyond EC2: Tips to Control Block Storage Costs

Get rid of orphaned volumes

The whole point of block storage (like AWS's EBS or GCP's Persistent Disk) 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. Before taking action, though, you should be sure you don't have any data retention needs for this data.

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 GCP'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 PIOPS, 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 IO 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.

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.

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.

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 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 buy RIs 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 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 prepay, 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 RIs or CUDs, 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 this in Chapter 14.

Unless you've reserved 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 reservations. Most cloud service providers allow some ability to exchange or change reservations to match new usage parameters. With some planning, you can avoid being locked into the incorrect reservations and start your usage reduction efforts by modifying reservations as you make changes.

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 only save 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 functions performed rather than on prepurchased units of capacity.

It 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. And 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.

But 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 lambda 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 is 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 success and profitability of a service. 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, GCP, 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 to 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 in order 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 using FinOps, you don't have to yell.) 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 running to an executive with a list of oversized resources, claiming widespread and rampant waste.

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 to investigate 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 have 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.

Your usage optimization workflow should allow for these reasons to be tracked. You can use a ticketing system like JIRA 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 and establish trust.

Crawl, Walk, Run

The Crawl, Walk, Run 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 Crawl stage, 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.

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

Here's a real-world example of how a Fortune 500 biotechnology company implements its Run stage 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-premise 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 11-3 shows optimization tasks assigned by owners with an associated color coding based on progress.

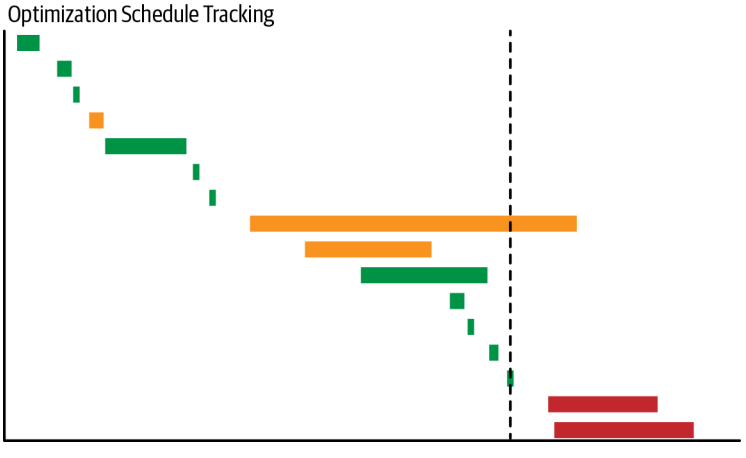


Figure 11-3. Schedule tracking of optimization tasks

The company's rightsizing process is fully automated. At the start, their FinOps team kicks off rightsizing scripts that link to the Apptio Cloudability 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 11-4) 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 11-5 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.

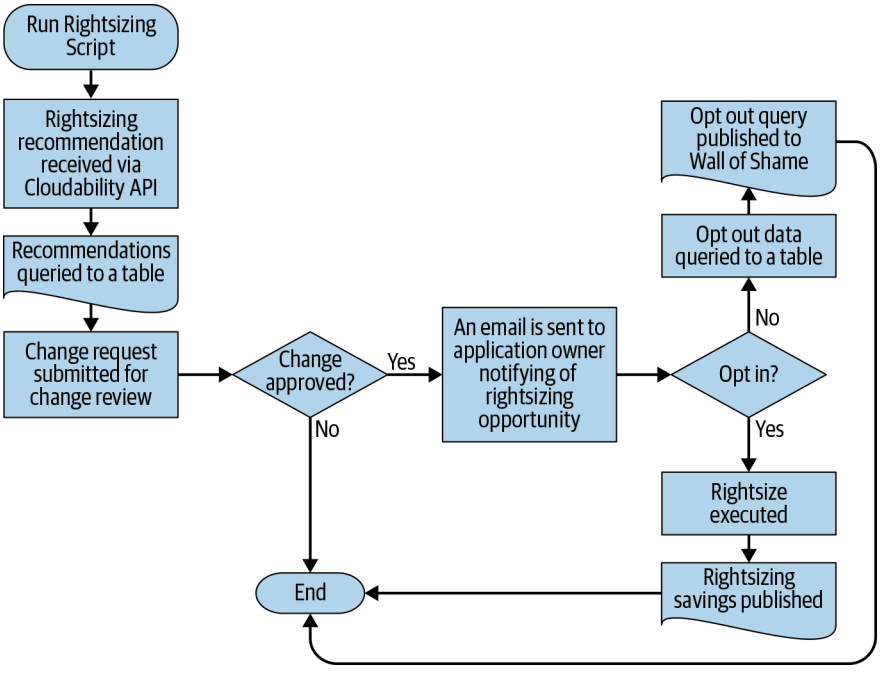


Figure 11-4. Automated rightsizing workflow

```
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 11-5. Sample email notification for automated rightsizing

Automating changes to an environment is an advanced (Run stage) 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

> 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.

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 due to 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.

> We use "mini business cases," which we generate when identifying opportunities for savings. These could be rightsizing, they could be service conversion (hosted DB versus managed DB, EFS to S3, etc.), they could be family migration (r3 to r5 instance family), or other items. We document them and track them over time to create a running ledger of recommendations and optimization opportunities. They're a key tool for the FinOps team to use to track and quantify their activities throughout the year.
>
> —Rob Martin, who does FinOps Practice Management at Apptio Cloudability

Some recommendations lend themselves easily to tracking their cost impact. For instance, r5 instances can save 45% over 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 r5s (the compatibility of EMR versions, 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 very 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 RI purchasing 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 sav-

ings 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 9), you can compare one team to another.

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.

Conclusion

Usage optimization is often more difficult than rate optimization, and 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 fewer resources when they're not needed.
- Visibility into waste can lead to a leaner culture inside an organization, as teams become more aware of the impacts of waste.
- 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 should be implemented carefully using the Crawl, Walk, Run methodology.

- 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 12

Paying Less: Rate Optimization

While usage optimization is about reducing resource usage to avoid costs, rate optimization is about paying less for the resources you continue to use.

By now, you know that not all cloud providers are created equal. With multiple purchasing options, billing in different time increments, and a slew of payment structures and commitments, each provider brings with it a variety of financial ramifications for customers. In this chapter, we'll cover some of the pricing options. Reserved Instances (RIs) and Committed Use Discounts (CUDs) are quite complex, so we'll save those for Chapter 13.

Compute Pricing

Depending on the cloud service provider and the resources you're using, there can be many ways to pay. 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 12-1.

Table 12-1. 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	Preemptible	Low-priority VM
Sustained use discount	N/A	Sustained Use Discount	N/A
Reserved	Reserved Instances/Savings Plans	Committed Use Discount	Reserved VM Instances
Volume discounts	Volume discounts	Volume discounts	Volume discounts

On-Demand

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 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/Preemptible/Low-Priority Resource Usage

Some cloud service providers offer a way to run a resource using the spare capacity of their cloud platforms. With spot/preemptible/low-priority pricing, you typically set a price (or a bid) you're willing to pay for the resource, which can sometimes be more than the on-demand rate but usually is set lower. You will pay the market rate for the spot resource until your bid is exceeded, at which point you will have a short period in which to stop using the resource before it's removed by the cloud service provider.

This pricing offers significant savings 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.

Reservations

As stated earlier, we'll cover reservations in great detail in Chapter 13. But generally, a reservation is a long-term commitment you make to the cloud service provider to run a set amount of usage for a period of time. In return, the cloud service provider will discount the normal on-demand rates for that set amount of usage.

Storage Pricing

When we talk about rate optimization with data storage in the cloud, we aren't referring to reducing what you store. That's usage reduction. Here we're referring to what tier of storage is chosen for your data. 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 and the quality of the data storage.

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, you call it *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 one 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 into 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 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 region level, meaning usage across regions won't be combined for discounting.

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.

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 12-1, 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.

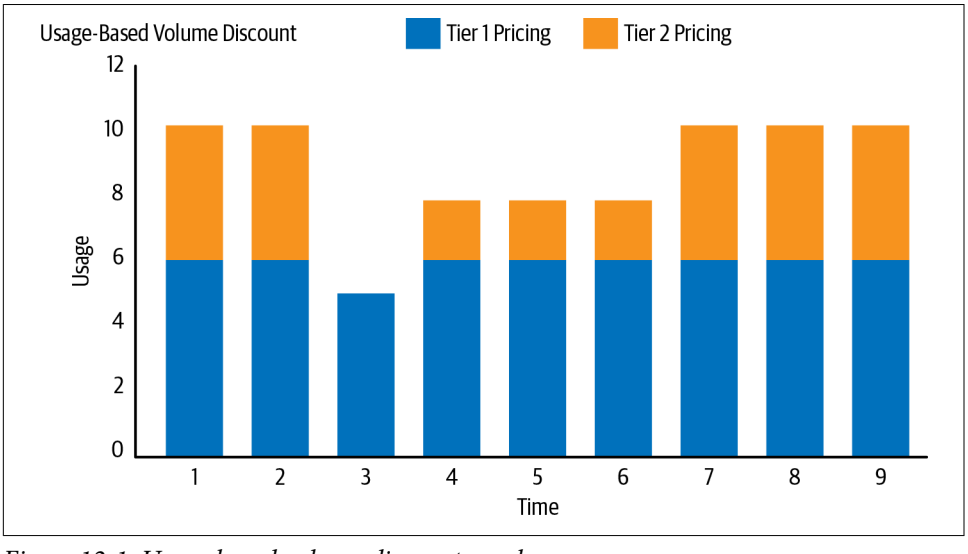


Figure 12-1. Usage-based volume discount graph

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 GCP Sustained Use Discount (SUD), where compute usage is charged at lower rates the longer it is running. 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 12-2, 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.

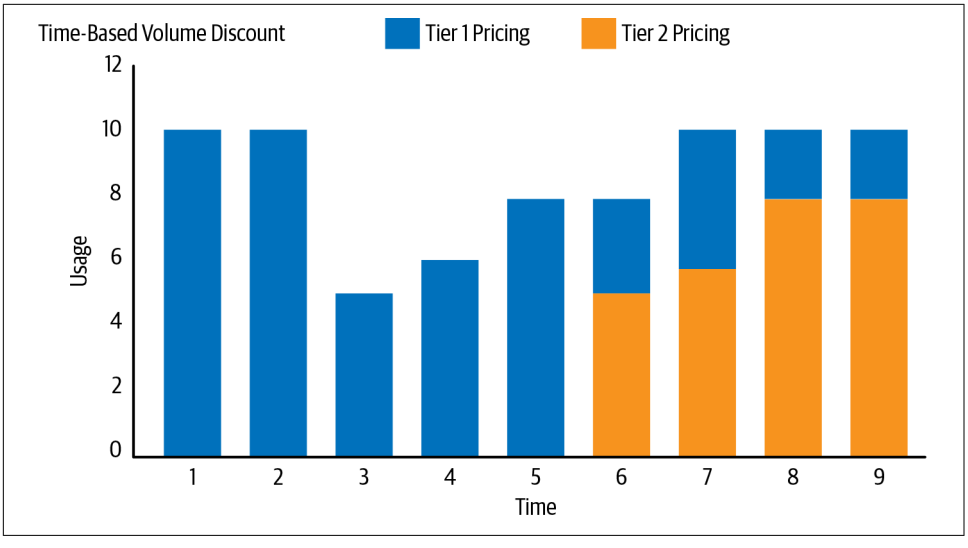


Figure 12-2. Time-based volume discount graph

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 and marketplace vendors for better pricing.

Custom Pricing Agreements

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.

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, GCP, 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 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 EAS), 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.

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 these two levers together can produce significant savings.

To summarize:

- Rate reduction is about getting charged lower rates for the same resources.
- By using the native cloud offerings correctly, you can reduce your cloud spend.
- Automation, whether it's offered by the cloud service providers or you run it yourself, can move your data and processing to lower-rate resources when appropriate.

- 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 method: using Reserved Instances and Committed Use Discounts.

CHAPTER 13

Paying Less with Reserved Instances and Committed Use Discounts

Reserved Instances (RIs) are the essence 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, RIs can bring a significant competitive advantage in the form of improved cloud unit economics, giving you the same compute power for a fraction of the price.

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 by going to *http://FinOps.org/Book*. Fear not, though: while there have been many changes to offerings over the years, the core best practices around reservations remain consistent.

Introduction to Reservations

RIs and Committed Use Discounts (CUDs), collectively known as reservations or commitments, are the most popular cost optimizations that cloud service providers offer. This is partially because they use extensive marketing to convince you that you can save massive amounts by going to the cloud, but it's also because FinOps platforms enable you to plan, manage, and benefit from these optimizations.

Each cloud service provider has a slightly different offering with its own specific rules on how it works and the discounts it provides. You must also consider the

implementation models that organizations use, based on their needs, and how the overall process should work inside an organization.

RIs 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 13-1, indicating that the company had cut its EC2 spending by 60% simply by purchasing RIs.

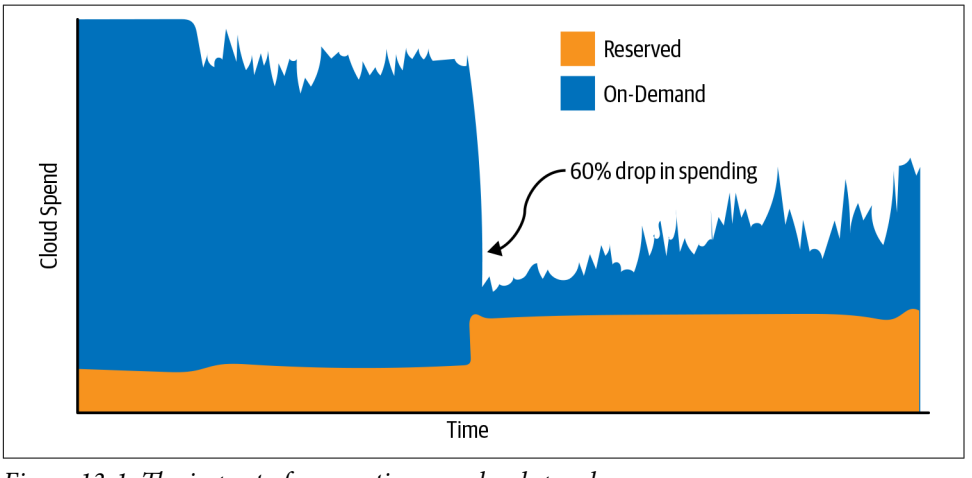


Figure 13-1. The impact of reservations on cloud spend

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 RI coverage is a process that takes a lot of organizational education and alignment.

Stories from the Cloud—J.R.

I worked with an FTSE 100 UK retailer that had millions of dollars a month of cloud spending but zero RIs. They knew that by buying even a modest amount of RIs, they could 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. Upfront payments for RIs appear to be capital expenditure (CapEx) spending that finance should depreciate as a physical asset. 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 dollars. 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.

Reserved/Committed Usage

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 75%.

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, RIs are essentially a billing construct. AWS RIs can offer some capacity functionality outside of the billing discounts, but we'll cover that more in "Amazon Web Services" on page 158.

RIs or CUDs are applied to individual resources in a nondeterministic 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. Providers will allocate the reservation randomly to a resource in the account. Randomly applying the discounts introduces many strategy and chargeback considerations that we'll discuss in Chapter 14. 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). 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 reservation, 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 reservation completely to save money.

The specific purchase options of a reservation and what resources they can apply to are different for each cloud service provider. Some offerings allow you to modify the commitment parameters during the commitment term.

Instance Size Flexibility

Cloud service providers offer hundreds of different server instance types, and reservations match particular resource usage parameters. This means you need to purchase potentially hundreds of specific reservations. To make this process easier, cloud service providers offer *instance size flexibility*.

Within a specific type of usage, 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 some vendors (AWS) exclude instance size flexibility from servers with licensed software, while others (Azure) don't allow you to discount the software license costs with reservations (such as applying a discount to the instance price only).

Due to the way GCP charges, they naturally do instance size flexibility by combining all instances' vCPU and memory counts within the same offering. Reservations apply to the combined total instead of against the individual size instances.

Conversions and Cancellations

A more recent addition to the reservation landscape is the ability to convert reservations or exchange them for another type. Convertible reservations typically offer a lower discount rate than reservations that cannot be exchanged. 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.

In general, the conversions you make must be equal or greater in value. 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. Separate to conversions, some cloud service providers (e.g., Azure) offer the ability to cancel some reservations, usually at a fee.

Overview of Usage Commitments Offered by the Big Three

Let's take a look at the big three cloud providers and how their offerings compare in Table 13-1.

Table 13-1. Comparison of reservation offerings across the top three cloud service providers

	AWS	Google	Azure
Program name	Reserved Instances/Savings Plans	Committed Use Discounts	Reserved VM Instances
Payment model	All upfront, partial, no upfront	No upfront	All upfront
Term	One or three years	One or three years	One or three years
Instance size flexibility	Yes	N/A	Yes

	AWS	Google	Azure
Convert or exchange	Standard, No Convertible, Yes	N/A	Yes
Cancelable	No; however, RI can be sold via a marketplace	No	Yes, up to a yearly value limit
Sustained use discounts	No	Yes	No

You can see that the big three cloud service providers, offerings are very similar, with the same 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 Reserved Instances 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, Elasticsearch 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 upfront 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*.

RIs are confusingly named, since you can't actually reserve an 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 a Reserved Instance Provide?

AWS reservations have two parts: a billing benefit and a capacity reservation for the type of resource purchased. The majority of RI purchases are made to take advantage of billing benefits. In some cases, companies may require the capacity reservation as well, when they expect to run out of capacity on a particular type of resource.

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 has offered on-demand capacity reservations, which allow you to perform capacity reservations separately from RIs. Note that when you're using on-demand capacity reservations in combination with RIs, it's essential to set your RIs as regional so the RIs can discount the capacity reservation.

Parameters of an 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-1 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-1A). If regional, the RI will apply to any availability zone in a region (e.g., US-West-1A or US-West-1B). Note that only zonal RIs receive a capacity reservation, but one can be purchased for regional RIs.

Availability zone (AZ)
: If a zonal RI, to which specific AZ should the RI apply (e.g., US-West-1A)?

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 (RHEL, MSWIN, Linux, 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 instance size flexibility will apply from the smallest to the largest instances, there's no current way to ensure that the discount gets applied to a specific resource instead of any other matching resource.

Linked Account Affinity

AWS Organizations provides a feature called *consolidated billing* that allows for accounts to be linked and costs to be consolidated into a single master payer account. There can be only one master payer account within an AWS Organization, and accounts can be linked to only one AWS Organization.

When you purchase RIs, you can perform this purchase at the master payer account or at any linked 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.

If the local account doesn't have any matching usage for a reserved instance, 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. 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. Disabled sharing effectively pins an RI to a linked account if it's critical for budget purposes that it not be shared, but it also could result in waste, as discounts won't be shared to others when not used. 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 13-2.

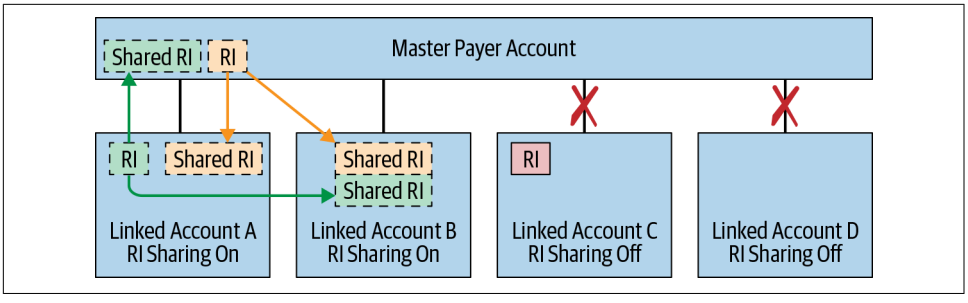


Figure 13-2. Discount interactions with RI sharing

Figure 13-2 shows a typical account structure: a master payer account with a handful of linked accounts. You can see in the diagram there are three RI purchases: in the master payer, in linked account A, and in linked 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 linked account via RI sharing.

In the example layout in Figure 13-2, the following may occur:

- RIs in the master payer 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 master payer account but will have preference for the RIs within the account.
- Account B can benefit from RIs from both account A and the master payer 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 Reserved Instances

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 a pro rata difference in any upfront 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.

EC2 RIs also contain a feature called *capacity reservation*. RIs configured as zonal (i.e., they apply to a particular availability zone) will hold server capacity available. This capacity is available only to the same AWS account as the RI and applies only to 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 availability zone in that same region, but it will not include any capacity reservation. Regional RIs can, however, be combined with on-demand capacity reservations to achieve the discounts of RIs across a region. But they retain the ability to reserve capacity in a particular availability zone and then have your regional RI discounts apply a discount to the capacity reservation.

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 13-3 illustrates a way of thinking about how ISF can be applied. Each column represents the instances that were run in a given hour. For r5s and c5s, 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 "points" at the smallest size within a family. This would mean you'd have only one actual RI purchase that's well matched to your usage. If instance sizes fluctuate but normalized usage stays the same or increases overall, your RIs will remain perfectly utilized.

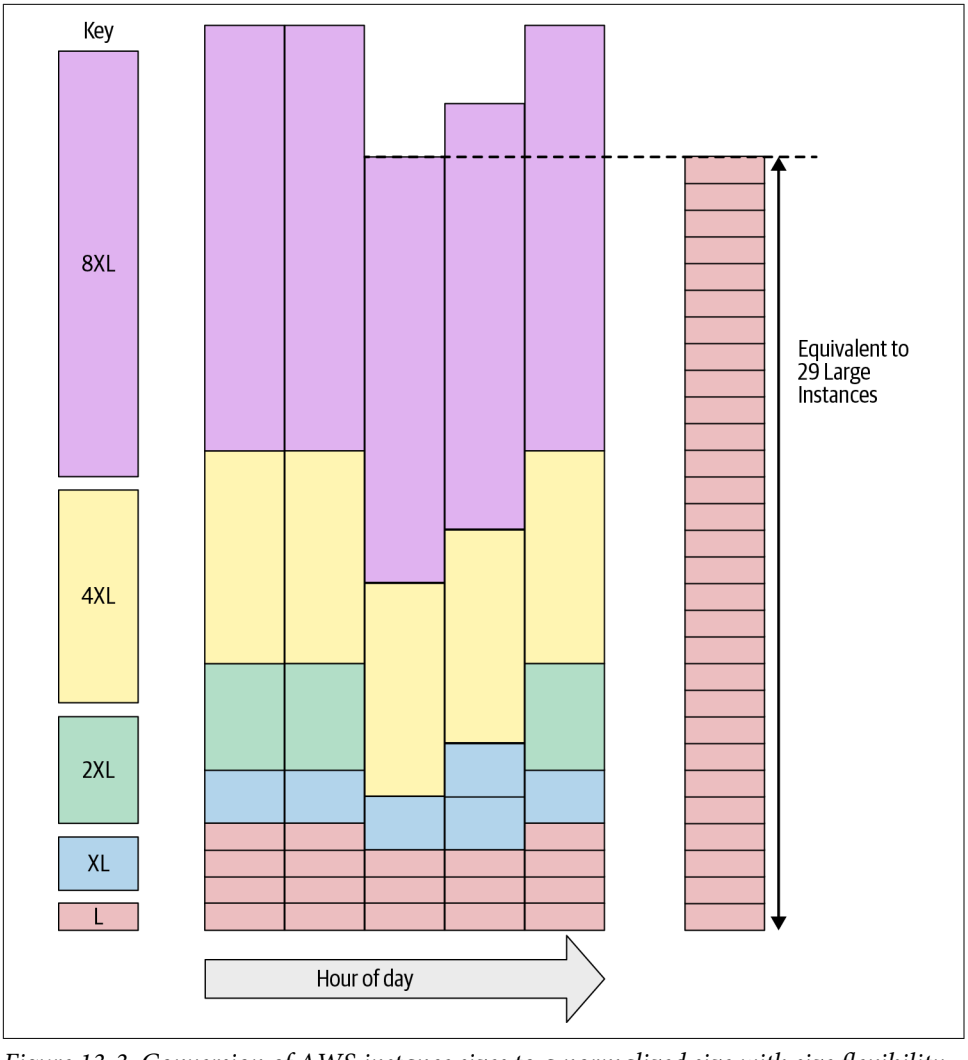


Figure 13-3. Conversion of AWS instance sizes to a normalized size with size flexibility

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 13-2 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 13-2. 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 13-4, you have four small reservations, which will cover four small instances. If you don't have any small instances with instance size flexibility, 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.

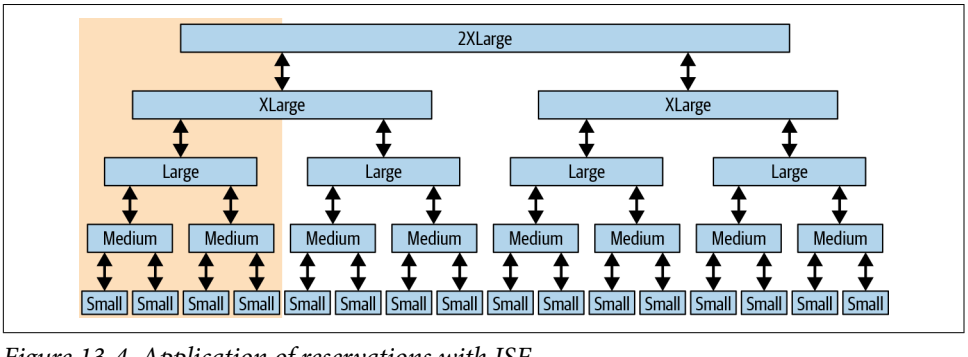


Figure 13-4. Application of reservations with ISF

Savings Plans

During the final stages of creating this book, AWS released a new commitment offering called Savings Plans. A large amount of what you learned here about Reserved Instances applies to Savings Plans as well. AWS has continued the purchasing options of "All upfront," "Partial upfront," and "No upfront" payment and one- and three-year durations.

Savings Plans are offered in two plan types:

Compute Savings Plans
: Apply broadly across EC2 and Fargate compute, offering savings comparable to convertible RIs. This plan type applies more widely than a convertible RI—even across regions—which will lower the amount of effort in maintaining high plan utilization.

EC2 Instance Savings Plans
: Apply to compute that matches compute usage more specifically, applying to a single family in a single region. While this plan type is more restrictive, it offers higher discounts and is less restrictive than a standard RI.

There are, however, a few major differences between RIs and Savings Plans:

- Both Savings Plan types offer more flexibility in the compute they apply discounts to, removing the need to consider tenancy, operating system, instance family, or instance size. Most important, Compute Savings Plans 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. Savings Plans are purchased as a committed hourly spend. Importantly, the spend amount that you commit to is post-discount. For example, if you're running $100/hour of on-demand EC2 and a Savings Plan offers a 50% discount, you need to purchase a $50/hour Savings Plan.
- Unlike RIs, Savings Plans apply discounts to AWS Fargate usage. This is the first AWS offering to discount usage across multiple service offerings.

While many of the RI techniques in these chapters will help you evaluate your Savings Plans decisions, it's too early for there to be specific best practices for Savings Plans. This is a perfect example of why we need the FinOps Foundation to continue the conversation, formulate the latest best practices, and share knowledge to all practitioners.

Google Cloud Platform

GCP infrastructure and services are different than those of the other cloud providers, with two unique billing constructs to simplify the cloud costs.

Let's look at GCP's zero effort Sustained Use Discounts (SUD) before getting into how Committed Use Discounts work. SUD is essentially a time-based volume discount. As you might remember from Chapter 12, the discount is applied based on how long the compute instances are running within each month. As your compute instance runs for particular percentages of the month, the rate you're charged reduces. The great thing about this model is it's zero effort. Just run servers as needed, and the cheaper

rate will start applying automatically where appropriate. The only drawback to the program is its lower savings rates as compared to the other rate reduction options.

For the best savings, GCP offers Committed Use Discounts, a GCP billing construct that requires committed usage offered at deep discounts. Contracts 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.[1]

As opposed to the Sustained Use Discounts, CUDs are not zero effort and do not automatically apply to your cloud bill. Prior to running your compute instances, you must make a purchase commitment for the amount of vCPUs and memory. It's very important to understand that this discount is applied not to a particular compute instance in your account but rather overall to your end-of-month bill.

Commitments are for a specific region and a specific project. Recall from Chapter 11 that projects organize all GCP resources into logical groupings. Zone assignment within a region isn't a factor like it is with other providers. Also notable is that discounts apply to the total number of vCPUs and amount of memory rather than to a specific machine or machine type.

Not Paying for VM Instance Hours

This might seem obvious, but the fact that GCP doesn't charge for virtual machine (VM) instance hours is a massive difference from the 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.

1 "Committed Use Discounts," Google Cloud, last modified November 14, 2019, *https://oreil.ly/1H-3K*.

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.

Billing and Sharing CUDs

Let's look at how CUD billing and application works within the Google concept of organizations, folders, and projects. Figure 13-5 shows the structure of the hierarchy.

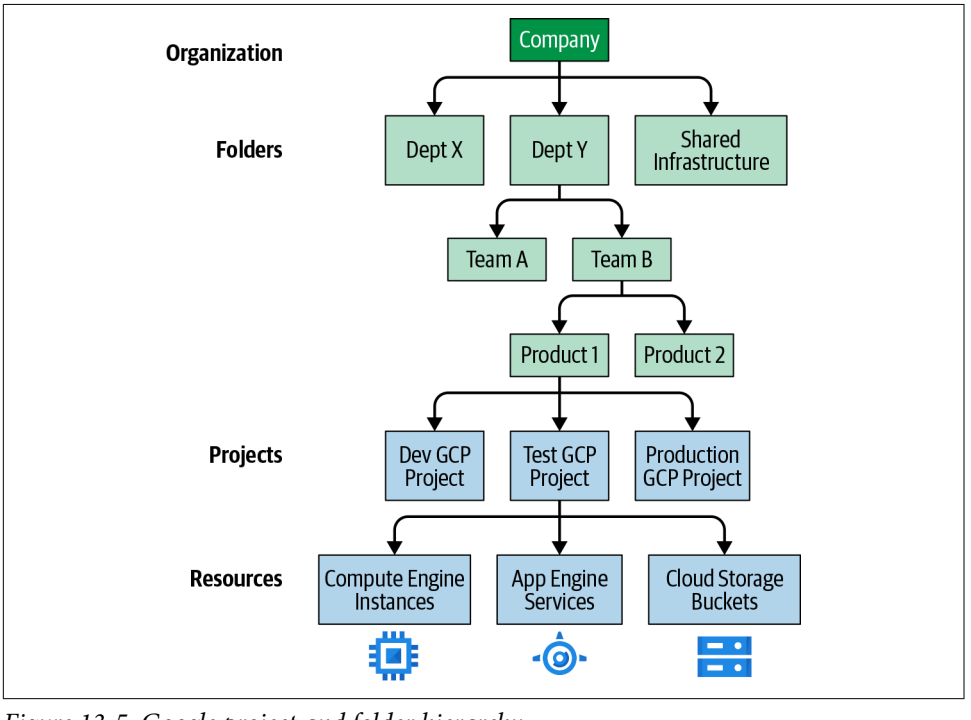


Figure 13-5. Google project and folder hierarchy

There are similarities to AWS's concept of a master payer structure. But even though the organizational account can own the CUD, the payment method and purchasing are done at the billing account level. While CUDs offer the flexibility of applying to a variety of machine types, they apply only to a single project.

Unlike AWS, where RIs are automatically shared across multiple linked accounts, CUDs can't be shared across projects. This could be seen as a pro or a con. If you want to allocate funds from a particular project to make the commitment, this will offer cleaner chargeback options than the way AWS handles RIs. However, CUDs could result in more waste than would be incurred in a model where unused

commitments are shared across multiple projects. CUDs essentially lock into the project for which they are purchased.

For an organization with a lot of projects, this lack of sharing means that when you do commitments, you can't take advantage of the economy of scale of the entire organization. You need to determine core counts at each project level.

Relationships Between Organizations and Billing Accounts

Two types of relationships govern the interactions between billing accounts, organizations, and projects: ownership and payment linkage.

Figure 13-6 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.

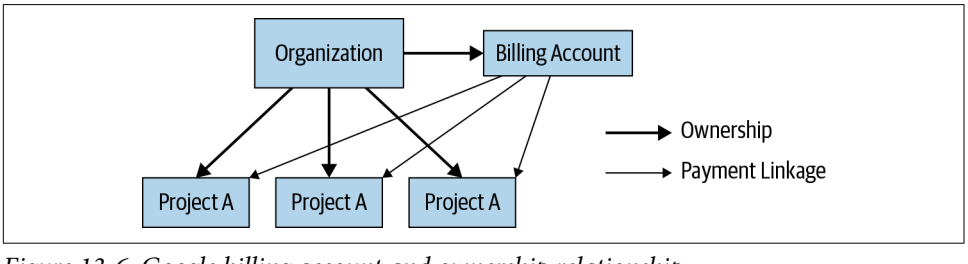


Figure 13-6. Google billing account and ownership relationship

Applying CUDs Within 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 in either general-purpose or memory-optimized commitments, and one type of CUD can't be applied to the other resource usage type. If you buy general-purpose CUDs, they will not apply a discount to memory-optimized instances, and vice versa. Not all GCP instance types are coverable with a CUD.

You purchase CUDs in one- or three-year terms, and they don't autorenew. You manage renewals by buying replacement commitments as old commitments expire. Other discount programs such as Sustained Use Discounts cannot apply to vCPUs/memory covered by a CUD.

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. Because no CUD was applied, SUD can apply to the remaining 12 vCPUs.

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.

Microsoft Azure

As long rumored and widely expected, Microsoft joined the RI party in late 2017, with an announcement timed right before AWS's big re:Invent conference in Las Vegas. Their offering is aimed at matching many of the 2017 AWS RI improvements, which added functionality like instance size flexibility and more convertible exchange options.

Azure RIs are purchased for an Azure region, virtual machine type, and term (one year or three years). And that's about it. Operating system and availability zone are not as specific as they are in AWS.

Azure RIs can be exchanged across any region and any series as a workload or application needs to change. However, these changes aren't yet programmatic and must be requested via a support form in the console. 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 possible to cancel Azure RIs 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 (as of early 2019) cancellations max out at $50,000 per year.

There is a capacity prioritization but no guarantee. This means that 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 the enrollment/account or subscription level, so you can manage RI usage at an organizational or individual department level. This gives you more control over how to apply RIs and who should benefit from them, and it solves the "RI float" dilemma in AWS, where there's little control over where RIs are applied. If the goal is to save money organization-wide, then RIs can simply be assigned at the enrollment/account level.

If a particular business unit wants to buy the reservation for its use only, the RI can be assigned to a subscription 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 linked 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.

By default, RIs are Linux-based, but Windows can be added either via Azure Hybrid Use Benefit (which assigns on-premise licenses to the instance) or via an hourly fee to add Windows (similar to AWS's model), 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 RIs are only "All upfront," requiring payment for the entire term at the beginning. There is no option to do a "Partial upfront" or "No upfront" reservation like there is in AWS, both of which allow you to take advantage of savings with less initial cash outlay.

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 in order to cover your VM instances.

Let's look at an example to help clarify how this works (see Table 13-3).

Table 13-3. 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 13-7).

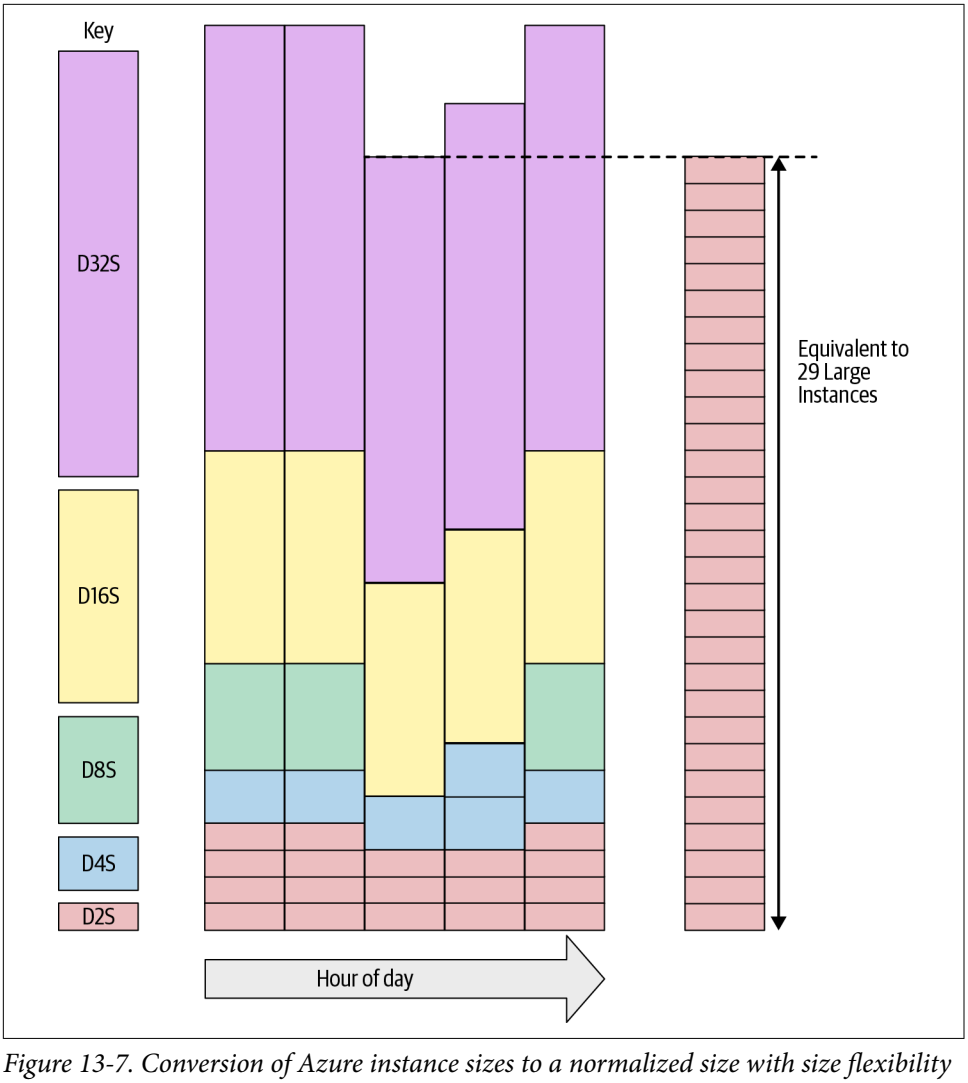


Figure 13-7. Conversion of Azure instance sizes to a normalized size with size flexibility

You can see from Figure 13-7 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.

Conclusion

RIs and CUDs are among 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 it works, and the discounts it provides.
- 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 must be sure to centralize the management of RIs 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.

Now that you know how reservations work, we'll move on to strategies and considerations for correctly implementing reservations inside your organization.

CHAPTER 14

RI and CUD Strategies

Now that you understand the fundamentals of reservations as described in the last chapter, it's time to ask some tough but important questions:

- How much should you reserve?
- What should you reserve?
- When should you reserve?
- Who should be involved in the process?
- How do you know your RIs and CUDs 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 reservation 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

There are a number of errors that we see companies make when purchasing RIs and CUDs. In no particular order, these are:

- Delaying purchasing RIs too long
- Making purchases that are too conservative
- Buying based on the number of unique instances instead of a waterline

- Not managing RIs and CUDs after purchasing them
- Buying too many or too few RIs or CUDs
- Buying the wrong RIs or CUDs

Nearly everyone gets some part of their early strategy wrong. That's OK. This is the Crawl 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

Steps to Building an RI Strategy

There are five key steps in building your first RI strategy:

1. Learn the fundamentals.
2. Build a repeatable RI process.
3. Purchase regularly and often.
4. Measure and iterate.
5. Allocate RI costs appropriately.

In this section, we'll cover each step in turn.

Learn the Fundamentals

Good news! Thanks to Chapter 13, you're mostly up to speed on the fundamentals of RIs.

However, there are a couple of concepts we didn't cover. The first is the break-even point of an RI. If you remember nothing else from this section, never forget this:

> When it comes to RI commitments, a year is not a year and three years is not three years.

Yes, you are committing to paying for that period of time, but you don't need to actually utilize the RI for that long 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 RI may break even in as little as 4–6 months, and a 3-year RI 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 RI break-even point.

Components of the RI break-even point

Figure 14-1 shows you the accumulated cost of a resource, for both on-demand and reserved, over a one-year term. In the example, the reservation cost over $300 up front, which is why the reserved line starts above $300. If you had used a "No upfront" reservation, then it would begin below the on-demand line. And if you used an "All upfront" reservation, it would be drawn as a flat line at the value of the upfront cost.

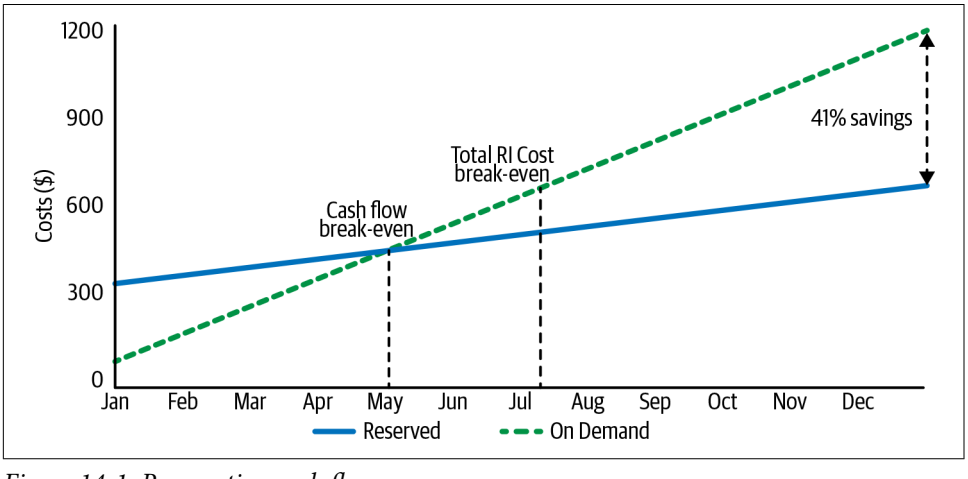


Figure 14-1. Reservation cash flow

The *cash flow break-even point* is the date on which the reservation 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 reservation at the cash flow break-even point. However, you will continue to pay the ongoing (hourly) costs of the reservation. 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 reserved 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 reservation and you would be no worse off than if you had not committed to the reservation. You would have paid the same amount on-demand versus with the reservation. After the break-even point, you realize savings from the commitment.

The difference between the on-demand line and the total reserved cost break-even point is the savings (or loss) made by the reservation. It's essential to understand that with this graph it doesn't matter whether you run one resource that matches the reservation 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 reservation applies a discount, you can then compare the total cost of the reservation 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.

Here's the takeaway: the majority of the time, a well-planned reservation will break even sooner than you expect. When it comes to RI strategy, the real challenge is typically less around deciding what to buy first and more around getting your organization aligned on how to buy RIs or CUDs.

The RI waterline

Before we move on to that process, there's another fundamental concept we didn't cover in the last chapter: the RI waterline. As you know, you don't buy an RI for a specific instance, nor can you determine where the RI or CUD will be applied. The reservations are applied nondeterministically to a resource that matches the criteria of the reservation.

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 14-2 shows some resource usage over time. If you purchased four RIs, they would each have 100% utilization; a fifth RI would have 90% utilization; a sixth RI, 80%; and so on. If the reservation saves you 25%, then, to save money compared to the on-demand cost, you would need to be utilizing the reservation more than 75% of the time.

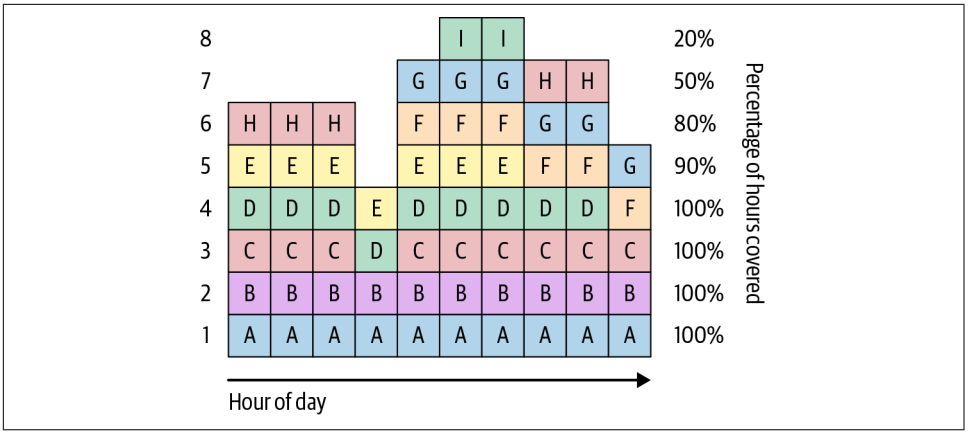


Figure 14-2. Reservation utilization

Knowing that, committing to seven reservations, you would be paying more for that seventh reservation than the on-demand rate, so any usage above six reservations isn't coverable.

In the rows above line 3, a single reservation 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 RI buying a core tenet of FinOps best practices. If you drive RI purchasing solely from the team level, you may end up with the wrong number of commitments. This is less the case in Google, where CUDs aren't shared between projects. Regardless, 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.

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.

Build a Repeatable RI Process

The second step in building your RI 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 RIs just when you need them: not too early and not too late.

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 RI 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 RI and CUD commitments to match. You may need to make monthly RI purchases to keep up with the rate of change, and you will almost certainly make more frequent RI exchanges, modifications, and adjustments to your portfolio.

Thus, there should be a dedicated person on your FinOps team who owns the RI commitment process. We sometimes call this person the *RI/CUD czar*. They need to have a deep understanding of how RI commitments work (see Chapter 13) and a sensitivity to the motivations of the teams running the resources against which they are making reservation commitments (see Chapter 3). They will need to interface with finance, tech, and executive teams to carry commitments over the line. Typically, this person is a technically minded business analyst on the FinOps team.

Why should you appoint a czar? The easy answer is that they can save your company 10–40% on your cloud bill. The more nuanced answer is that the job of deciding when and how many RIs to buy is typically not done well by tech or ops teams, who are focused on running their resources efficiently. Remember that you want to create a division of duties, where the FinOps team optimizes for rate reduction while the product or application teams optimize for usage reduction.

Purchase Regularly and Often

Cloud is designed for just-in-time purchasing. You shouldn't be running infrastructure or purchasing capacity (i.e., buying RIs) in a data center fashion.

Years ago, AWS had graphs like those in Figure 14-3 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 RIs.

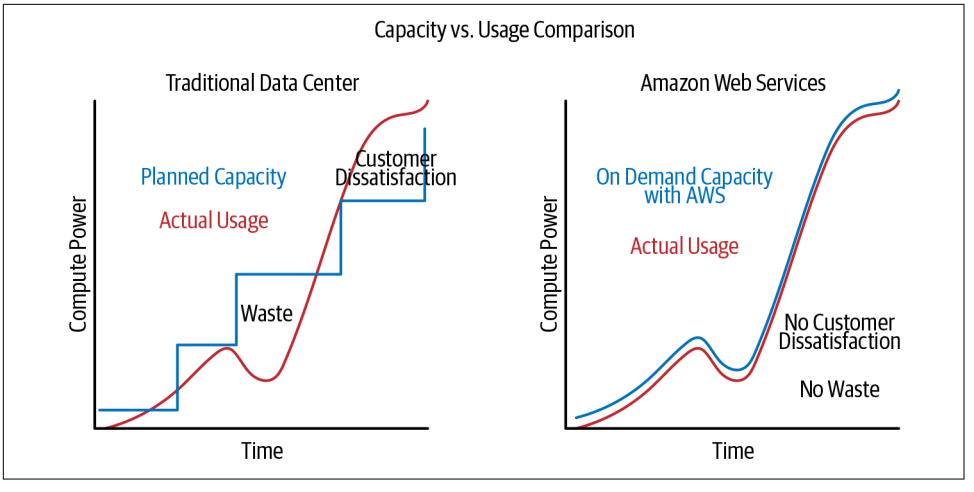


Figure 14-3. Diagram AWS used to host on their website

Measure and Iterate

We have heard many horror stories of companies that have purchased RIs incorrectly. To avoid this common mistake, make sure your initial RI purchases are small. You should have carefully set metrics with thresholds. These can start with simple measures, such as what percentage of resources that should be reserved *are* reserved. Conversely, you should track the amount of waste (or vacancy) of existing reservations to ensure they are used to their maximum extent. For more on the metrics to use in managing RIs, we cover metric-driven cost optimization in Chapter 16.

As confidence grows in your process, you can increase both the frequency and the size of your purchases.

Allocate RI Costs Appropriately

In Chapter 4 we introduced the matching principle, which holds that expenses should be recorded in the period in which the value was received. 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 upfront payment over the term. Distributing the upfront payment is called *amortization*. With some cloud service providers like AWS, you are given the option of paying all, some, or nothing up front. You receive a higher discount for the reservation the more you pay in advance.

Deciding whether or not to pay up front requires two major considerations:

Accounting
: The first is the impact to your accounting processes. We highlighted in Chapter 13 that RIs will apply randomly across your cloud usage. If you need to allocate part of the upfront payment to one team and another part to another team, you will be required to track exactly where the reservation applied its discounts and then amortize accordingly.

Granularity
: You can't just divide a reservation into equal monthly parts. Not all reservations 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 reservations later in the chapter.

The Centralized Reservation Model

Successful FinOps practitioners centralize the reservation buying process. Reservations are complex, and you need to have an expert in control (the aforementioned RI czar). Having one team responsible for the analytics, approvals, purchases, and management of all reservations within your organization will ensure efficiency and accuracy, enabling the highest levels of savings. If your teams individually analyze their own use of the cloud, they often get it wrong, or they may double up on purchased reservations.

Reservations managed by an individual team take into account only that team's usage. The peaks and troughs in usage that an individual team considers when buying reservations will exclude usage that might well be coverable. While one of your teams reduces resource usage, another team could be increasing usage at the same time, resulting in an overall flat usage. Individually, there appears not to be enough usage to require a reservation. But the combined usage is consistent, and a reservation could save your organization money. A centralized view takes into account all teams' usage and will result in an overall higher savings rate for your organization.

It should be clear by now that reservations are complex, and as each cloud service provider continues to evolve its offerings, it's very likely they will be even more so. Having individual teams around your organization taking into account the changes in offerings and updating their own analytics and purchasing processes is inefficient. It will lead to teams performing reservations incorrectly and will ultimately cost the business more.

To reduce the demand for individual teams to manage their own reservations, 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 reserve their own usage in order to gain savings. Having distributed teams buy reservations when you're operating in a centralized reservation model is dangerous, as you're likely to end up overcommitted.

A centralized reservation 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 reservation can be converted or exchanged, less input from individual teams is required, as the FinOps team can convert the reservation when needed. If a reservation 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 reservation into account when planning any change to the resource usage in the future.

Timing Your Reservations

We've heard many opinions on how companies should time their purchases of reservations. It's important to understand the impacts of committing too early versus committing too late. If you reserve earlier than needed, you'll be paying for the reservation without any discounts being applied to your account. If you reserve more usage than you end up needing, you will be paying for the reservation 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 14-4 shows some resource usage in a cloud bill. If you purchase a one-year reservation in January, you will end up paying for the reservation for three months without obtaining discounts, wasting 25% of the RI (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 reservation doesn't save you more than 25%, then you will end the year paying more than you would have had you not made the reservation at all.

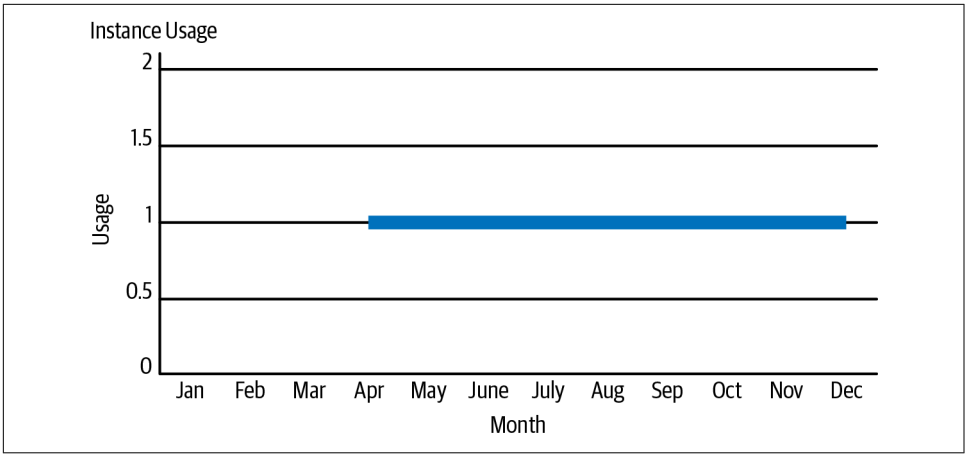


Figure 14-4. Resource usage starting in April and running for nine months

Inversely, if you had waited until September to make a reservation, 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 reservations only when needed in Chapter 16. Most companies start out making their reservations 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 reservations yearly might commit to 30% more reservations than they need. The idea is that as the year progresses they have an increased cloud spend, and the spare reservations will start applying a discount immediately. However, during the first part of the year, the company is overpaying for these extra reservations that are not applying a discount. As shown in the previous example, there is a risk that these reservations never apply more discount than the amount of unused reservation costs.

By increasing the frequency at which you perform your reservations, 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 reservations. There are a couple of roadblocks to performing reservations at a higher pace, however. The time it takes you to analyze the amount of reservations 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 Reserve

You saw how rightsizing impacts reservations in Chapter 11. For this reason, many people believe that the best approach is to rightsize first and reserve later. As correct as this approach sounds, it is not the right one. It invariably leads to lost savings opportunities through unnecessary on-demand spending.

Rightsizing is hard and can take a long time. 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 reserve 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.

One of the biggest mistakes companies make when their FinOps practices are immature is waiting until they have "completed" rightsizing before they reserve anything. Remember, you progress through the FinOps lifecycle many times, each time making the most critical optimizations.

The Crawl stage of usage reduction may be getting rid of unattached storage volumes, or turning off 100% idle machines in dev/staging accounts. The Crawl stage of rate reduction may be buying your first 10% of coverage (maybe even with "No upfront" RIs if you're using AWS). Getting to Walk and Run on 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 reservations, especially if your cloud usage is growing over time due to migrations or new applications.

When starting out, you should set a small reservation coverage goal, maybe 25–30%, and reserve some portion of your cloud resources, thereby realizing savings early. As you reduce your waste, your newly purchased reservations 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 to reservations.

Since you're able to measure the amount of waste in your accounts, you can reassess your coverage goal and ideally increase it over time. The Walk stage will see you getting on a regular cadence of buying RIs, while the engineering teams regularly clean up unused resources and rightsize underused ones. The Run stage comes when we get to metric-driven cost optimization (Chapter 16).

Building Your Strategy

No matter which cloud service provider you use, the goal for reservations/commitments is to save as much as possible. Before you develop a reservation strategy, there are a few things to consider. Your company's profile will affect the strategy of your reservations. You also need to consider the level of 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 upfront payments.

Level of Commitment to Your Cloud

When you create a reservation with a cloud service provider, you're committing to spend a certain amount of money with them. There may be an upfront charge and an hourly charge for that commitment. When you combine all charges over the full commitment term, you get to the total reservation 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 reservations, 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 reservation. The extra savings you receive won't be realized if you stop before the full three-year term.

The Cost of Capital

While reservations save money, they do require available cash to apply to upfront 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 upfront reservations 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-upfront 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 RIs.

A consumer's WACC 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 Goal Seek—that is, adjusting inputs to achieve the required output—exceeds the client's WACC, the client should purchase "All upfront" reservations. If the figure is lower than the consumer's WACC, then the implied interest rate of the "No upfront" RI purchase option will be less expensive to the consumer than the "All upfront" option.

The Red Zone/Green Zone Approach

In most cases, reservations 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 13. With this in mind, reservations aren't often purchased in low-use areas. Buying reservations in a region with low use has a risk of being uncovered, therefore costing the business instead of saving. In the same vein, purchasing reservations for older-generation instance types has similar risks.

When teams build infrastructure in these regions and/or use instance types that are deemed too risky to reserve, they won't gain any reservation discounts. Dave Van Hoven, Principal Business Analyst for Cloud at HERE Technologies, has an approach to building visibility into these reservation black spots, which allows engineers to make informed decisions about how they build.

The *red zone/green zone approach* is to publish information internally to the business on locations and instance types that will be automatically factored into reservation purchases by the centralized FinOps team (the green zone), and locations and

instance types that won't be covered by reservations without asking the FinOps team (the red zone).

The green zone
: The green 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 reservations in their day-to-day considerations. This green zone helps drive engineering decisions into areas that the business wants most to support.

The red zone
: The red zone requires engineers to either manage their reservations themselves or work closely with FinOps to purchase specialty reservations. Any changes in the engineering team's infrastructure will always require them to consider the extra commitments they have made to reservations, as it will be unlikely that the reservations can be modified to be used by another team. The red zone requires extra effort, and in turn becomes less attractive for engineering teams.

The black zone
: A small extension of this concept is the black zone, which clearly states which locations and types of reservations the organization does *not* approve of purchasing. This can be used to delineate where engineers shouldn't be building infrastructure.

The red zone/green zone approach makes it clearer how a centralized reservation practice works. It shows teams where reservations will not automatically apply to them and defines a process to apply reservations to obscure usage. Reporting should be provided to highlight the efficiency and savings being generated not only by the centralized FinOps reservation process (in the green zone) but also by the red zone.

Purchase Approvals

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 reservations monthly would be troublesome. This is where FinOps can benefit the business by building out new processes for performing reservations, with the understanding that improving the process saves the business even more.

We've seen successful companies allocate a quarterly budget to the FinOps team for reservations, with additional approvals needed only if the overall commitment is over this budget. In most cases, reducing the number of people and the amount of paperwork involved in the approval process speeds up the reservation frequency. We'll talk more about metric-driven cost optimization in Chapter 16, where we'll see how having a preapproved spend amount can lead to automated reservation purchases.

Who Pays for Reservations?

Approving a FinOps team to *purchase* reservations doesn't necessarily mean they will *pay* for the reservations. There are two main components of the cost of a reservation: the upfront fee and the hourly fee. Some reservations consist of only one of the two: "All upfront" and "No upfront" reservations.

With any upfront payment, the cost should be amortized out, since the reservation is applied to your cloud billing. Any unused capacity of the reservation is then charged against your bill directly and left unamortized from the initial upfront fee.

Once the cost is amortized correctly at the end of the month, you'll end up with:

- Prepaid expense accrual of the remaining upfront 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 reservation charged centrally in the billing data

When a FinOps team is approved to spend on reservation purchases, they add money to the prepaid expense accrual. This isn't a business expense as far as monthly costs go. In accounting, this is a business 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 reservation purchases as a business expense.

When costs are amortized out to your cloud resources using tags, labels, and accounts, you can attribute the reservation costs to your showback/chargeback cost allocation process and handle them as you do all of your other cloud costs.

For the unused reservations, 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 reservation cost:

Account-based attribution
: If reservations are purchased inside team accounts—close to the usage to which they are intended to apply discounts—it's possible to distribute the unused reservation costs, based on the account in which the reservation is purchased. The charges are allocated to each team owning the account.

Blended amortization
: Adding the unused reservation cost to the amortization rate effectively blends the unused cost into the rate everyone pays for resources covered by that reservation. This leaves uncovered resources at the normal on-demand rate. The variability of which exact resources a reservation 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 difficult. We recommend using resource-blended rates when performing showback/chargeback.

Resource-blended rate
: Here you create a new rate for resources by blending the total costs of resource rates across the bill. We aren't referring to the AWS billing data concept of a "blended rate," which we discussed in Chapter 4. Instead, this concept takes all usage matching a reservation (on-demand and reserved), applies the upfront amortization and unused reservation 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 reservation was applied in the billing data. This essentially creates a company's own averaged rate—in between the on-demand and the reserved price—for each resource type that changes as reservation coverage does. It avoids having "random winners and losers" wherever a reservation applies.

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 reservation rates for their usage, and the unpredictable application of reservations to usage is no longer an issue. All savings are reported centrally to the FinOps team, from which the unused reservation 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.

The approach you take to assigning costs of reservations will depend on your business processes and possibly will change over time as your FinOps practice matures.

Strategy Tips

You might find the following pointers useful when building out your reservation strategy:

Build visibility first
: A story we've heard time and again goes something like this: "I purchased a reservation, but my bill just went up." Often reservations 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 reservation is able to make, and without any visibility into how much a reservation is saving, all you have is an increasing amount of cloud spend.

Understand how reservations work

From Chapter 13 you should have an appreciation of just how complex reservations are. The most common mistake people make when buying reservations is not understanding them correctly and then buying the incorrect type of reservation, or buying them in the wrong locations. When a reservation 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 reservation program will be most likely to succeed. If reservations are new to your organization, you should consider using an experienced contractor to get started.

Having one person try to build all of your billing data processing, reservation recommendations, and reporting is only going to delay a reservation purchase. Most likely, the delay in building these recommendations and reports will prevent more savings than the cost of a FinOps platform. Also, these third-party platforms create reports that are maintained by teams of people with years of experience and have been validated across billions of dollars of cloud spend.

Avoid going from zero to hero

When you first start the Crawl stage, we recommend buying just one reservation. This allows you to confirm your knowledge with a low-risk purchase. You can verify that all of the common mistakes have been avoided before you make a more substantial purchase. Trying to go from "zero to hero" often leads to failure. You can consider the red zone/green zone approach we discussed earlier, identify the lowest-risk usage that could benefit from reservations, and then purchase in increasing amounts. It's a great way to build confidence in your reservation strategy.

And 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 overcharging by a cloud service provider. Once again, you find and purchase some low-risk reservations and then continue learning and growing your reporting processes as you start to see those savings.

When starting, we recommend setting reservation goals low. Consider 25–30% reservation coverage in the green zone. As confidence grows, revise the goals

upward. A common coverage amount we see organizations aspire to is 80% reservation coverage.

Consider the overall impact
: A reservation is going to apply somewhat randomly to a cloud bill. So if you have existing cost reporting and teams tracking spend closely, a reservation purchase can cause confusion when spending drops in one area but not another. Communicating the purchase of reservations, 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 reservations.

Conclusion

FinOps strategies for managing reservations vary from organization to organization. But there are a few common practices that lead to efficient and well-managed reservation 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 reservations work and spend time building visibility before your reservation purchases.
- You shouldn't delay buying reservations. Purchase low-risk reservations first, and then build on your processes in order to drive up coverage.
- Communication is important to your entire organization to avoid confusion as to who is doing what.

A correctly implemented reservation strategy will evolve and mature with FinOps, saving an organization large amounts. Ideally, metrics are used in all FinOps processes.

This concludes the optimize phase. 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 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 explore the role automation plays in FinOps, work through operational processes for success, and discuss the ultimate FinOps model: being able to track your costs against business function with the unit metrics reporting model.

CHAPTER 15

Aligning Teams to Business Goals

Without a process for action, applied 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.

With each iteration of the lifecycle, another goal comes into focus, with new processes and automation being built upon in order 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 in order to correctly measure each change you make.

It's becoming common for organizations to hire experienced FinOps practitioners to assist in solving their cloud financial management problems. Alternatively, there are FinOps consultants, from consulting partners and systems integrators (for example, PricewaterhouseCoopers, Deloitte, and KPMG now offer advisory services in the area) to cloud financial management platforms (especially those specializing in FinOps practice management) and independent contractors offering years of experience from working with many organizations. The FinOps Foundation has a certification process, which enables you to identify experienced, knowledgeable partners and train new hires.

Over the years, many organizations have developed processes around their FinOps practices. While workflows differ between organizations, there are many common elements to the processes that successful FinOps practitioners follow. We started the FinOps Foundation so that FinOps practitioners could share recipes for success.

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.

Processes

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 ("owns") the process, what parts of the organization must follow the process, and to what goal or goals the process contributes. A clearly defined process outlines expectations for staff.

When you adopt third-party automation tools—which we'll cover in Chapter 17—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 of when processes should be followed and the outcomes of following the processes (*visibility/alerting*), and finally the process that should be followed (*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 happens with cloud growth; otherwise, you end up with a process that works for the present but not for the near future.

At Atlassian, there are specific processes for onboarding new 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.

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 reservations), while others are distributed (like rightsizing).

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 performing the Crawl, Walk, Run model of 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 theater lighting design, there's a maxim that reminds the lighting technician to move only one light at a time—that way, they can see what changed. The same holds true here.

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 IT 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 running a FinOps team successfully when building processes such as rightsizing and RI 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 11, we highlighted that recommendations are 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 an RI 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 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.

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.

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.

Working with Bad Citizens

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.

As the goal of $50,000 in costs avoided is reached, new—and hopefully more ambitious—goals can be defined.

Conclusion

In order 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.

- Crawl, Walk, Run 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 bad citizens not completing their required tasks.
- Collaborate with FinOps practitioners outside your organization in order to adopt processes already proven to work at other organizations; consider joining the FinOps Foundation, using FinOps contractors, or hiring experienced practitioners.

Automation in FinOps should follow only a clearly defined process. In the next chapter we discuss an advanced (Run stage) FinOps process that enables just-in-time optimizations using metrics and alerts.

CHAPTER 16

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. When you are buying RIs, you're in the operate phase. With MDCO, you want the metrics to drive you.

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 a massive billing file 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 10 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.

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 *reserved 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 walk through reserved coverage to explain this in more detail.

Reserved coverage

Reserved coverage is the metric that shows how much usage is being charged at on-demand rates versus reserved rates. Generally organizations set a target coverage—commonly, it's 80%. Let's compare the various methods.

In the traditional model, you measure coverage by counting up all the reservation hours that you have and then dividing that number by the total hours you ran in the period (see Figure 16-1).

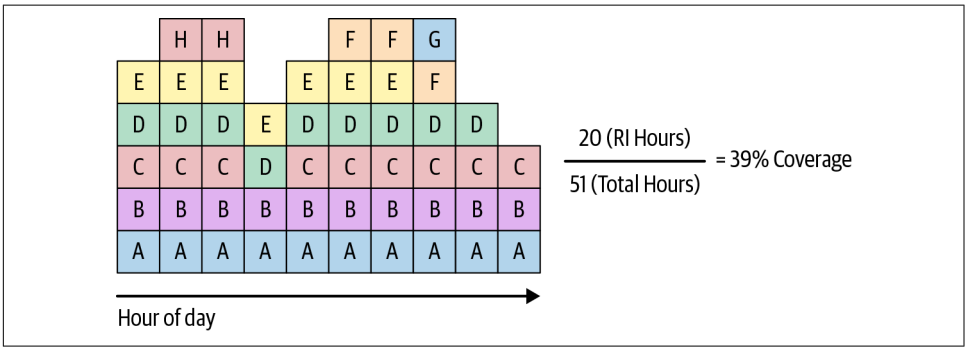


Figure 16-1. Reserved coverage calculated via raw hours

Even though the way coverage is measured in Figure 16-1 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 reservation because there simply isn't enough usage during the period of time to warrant it. Ideally, you don't include all the compute hours that are under the break-even threshold, below which reservations would cost more than they save. By removing these noncoverable hours, you can measure coverage over coverable hours (see Figure 16-2).

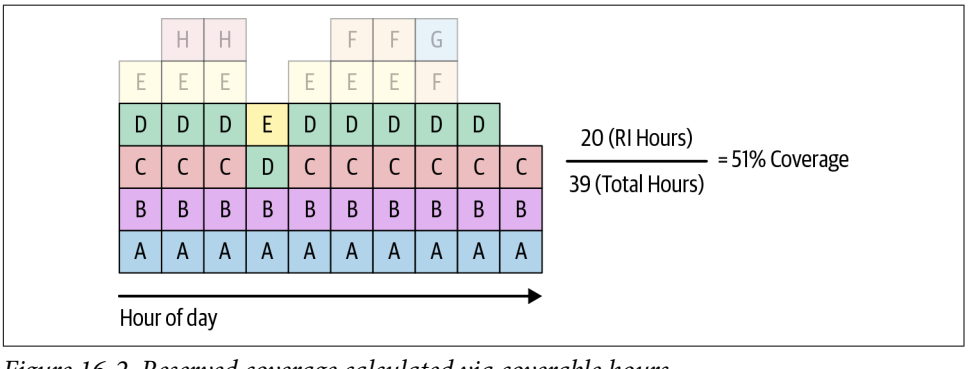


Figure 16-2. Reserved coverage calculated via coverable hours

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 16-2), you get a clearer picture of the true coverage percentage. A and B are covered by a reservation, leaving the hours by resources C, D, and E at on-demand rates. Buying an additional two reservations provides 100% coverage, and the two extra reservations 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 instance. Instances marked as A may cost over $1 an hour, while B may cost 25¢ an hour. The amount you save on each reservation would then be vastly different. Primarily, you buy reservations 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 reservation clearly shows the effect of a reservation on your cloud bill. Adjusting the graph by the relative savings of each instance type, you end up with a very different picture (see Figure 16-3).

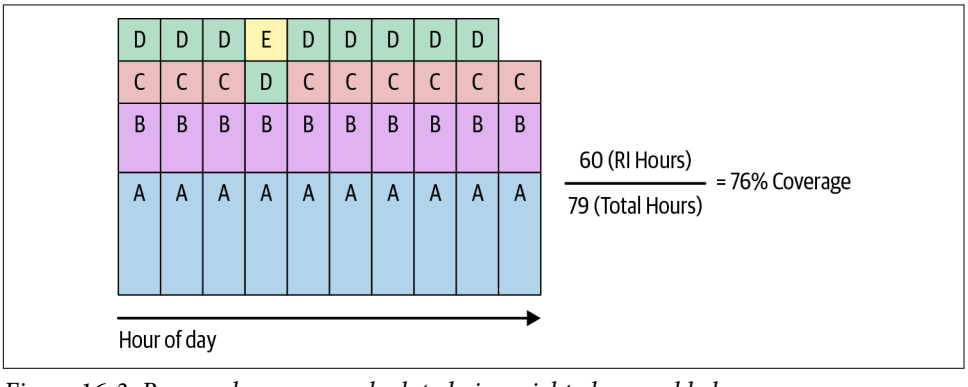


Figure 16-3. Reserved coverage calculated via weighted coverable hours

Figure 16-3 shows that you're actually closer to 76% of achievable savings. The original assessment in Figure 16-1 showed you as being in a very low-coverage situation. But the new weighted position, which also removes hours that shouldn't actually be reserved, shows that your position isn't really that bad.

The key takeaway is that reserving different resources realizes different amounts of savings. Not all usage is coverable, so including it all in reserved 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 reserved gives you a concept of *total potential savings*. For example, let's assume all of the cells in Figure 16-3 would save $102,000 per year when covered with a reservation (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 instance hours and reserved 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 RIs and enables conversations within the organization. You've created a common lexicon for mutual understanding.

Combining metrics

To determine a performance metric for your reservations, you measure how much of a reservation is being used versus not being used over a specific time period. This is called your *reservation utilization*. At an individual reservation level you monitor by time, measuring the proportion of hours (or seconds) that a reservation applied a discount to your resource usage, versus the amount of hours (or seconds) you didn't have any usage for the reservation to discount.

However, if you roll your reservation utilization up to show one overall performance metric, the individual underperforming reservations can be hidden. Instead, you measure individual reservations and alert when they individually underperform. Let's look at an example of this in Figure 16-4.

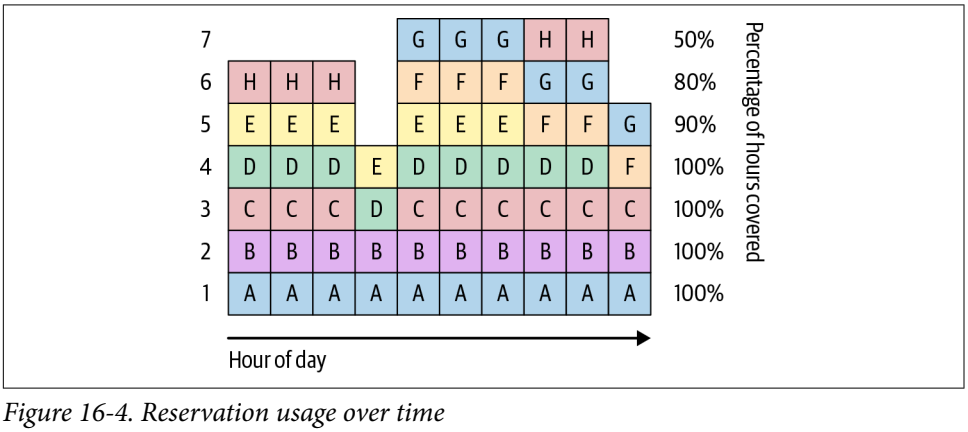


Figure 16-4. Reservation usage over time

You use the usage in Figure 16-4 to determine your reservation utilization, based on having seven reservations. You can see that the first four reservations are utilized 100%. The fifth reservation will have 90% and the sixth will be utilized 80%, with the last reservation 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 reservation 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 6. 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.

Metric-Driven Versus Cadence-Driven Processes

In Chapter 14, we discussed strategies around RIs. Most companies, when starting out with reservations, will purchase their RIs 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 16-5 shows some usage over time, and in dark gray you see the reserved coverage you have. You can see from the graph that purchases of RIs are occurring on a relatively set cycle, and between these purchases resource usage continues to grow. By buying RIs more frequently, you can achieve greater savings.

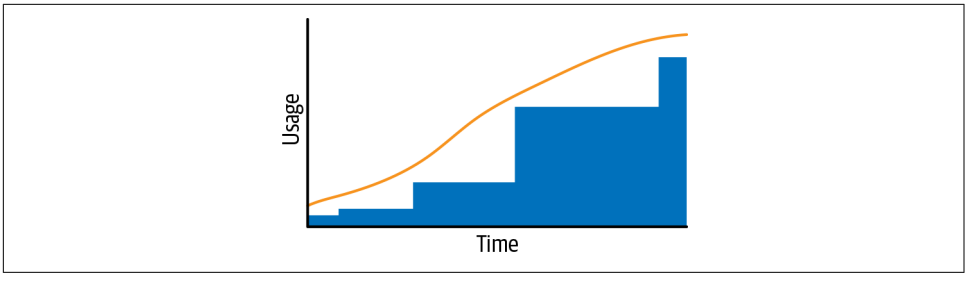


Figure 16-5. Reserved coverage when performed infrequently

Figure 16-6 shows a much more frequent RI purchase program. The frequency isn't always the same over time, but the resulting reserved coverage follows the resource usage growth tightly.

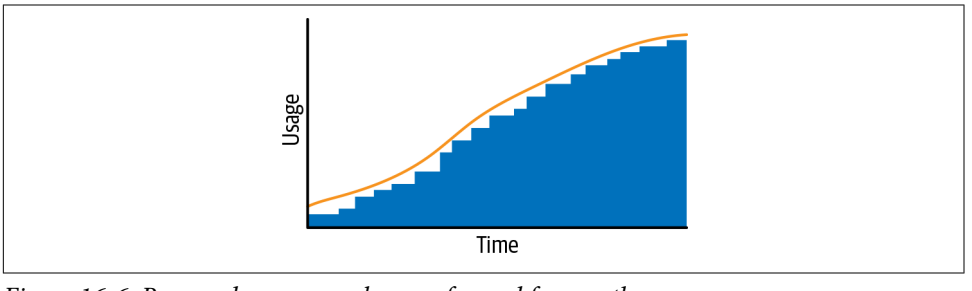


Figure 16-6. Reserved coverage when performed frequently

Schedule-driven cost optimization is fine to get started, but continuing to perform purchases slowly leaves savings on the table. However, looking at reservations 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 reserved 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 reservation 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 reservation to get the most possible savings. When utilization drops below 90% for an individual reservation, it triggers the computers to create a ticket, telling me that changes are required. At that point, modifications or exchanges of affected reservations 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 to action an MDCO alert, what the correct approval process is for things like buying new RIs, and what the correct order of operations is for modifying and exchanging existing reservations.

Have a preapproved RIs purchasing program within your organization. Giving the FinOps team the freedom to buy reservations as needed—up to a certain amount per month—reduces the lag time between an RI coverage alert and the purchase of more RIs to cover the shortage. Improving processes helps reduce the delay in taking action.

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.

There are many processes inside MDCO that require automation: report generation and alerting are two primary candidates. Within FinOps there are plenty of other tasks that can be automated. We'll discuss those next.

CHAPTER 17

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.

What's the Goal of Automation?

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.

What Is the Outcome You Want to Achieve?

It's important to identify an actual 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 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 (discussed shortly) 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. 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. In cases like this, you seek to validate that automation will help you deliver 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. For example, 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 adding supplementary tag data. Teams tag their resources with a single identifier and then have automation add extra, clarifying tags. For example, teams might use a

project identifier that maps to cost centers or to environment types. Automation would find that project identifier and then add the extra tags to the resources. 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.

Being able to have a conversation with the business that's not just about using 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 benefits over time. 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. 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.

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 HIPAA, SOC2, or PCI, and the third-party tooling providers may not currently hold the right certifications.

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 the 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. 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.

Automation Working Together

Why have just one automation tool when you can have two?

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.

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 our Crawl, Walk, Run 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 do deploy is working effectively. Here are some tips:

Use it in an inform mode first
: Start initial automation in an inform mode, 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.

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

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

In Chapter 11, we covered items like resource rightsizing and idle resource detection. Reduction automation removes such waste, or at least sends alerts to responsible staff members in order 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:

- 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.
- You should always 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.

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 18
FinOps for the Container World

Alongside the adoption of microservices, containers have gained popularity. Over the last few years, the number of concurrent containers being run by organizations has rapidly increased. Managing a single running container instance is quite simple overall; however, running hundreds or thousands of containers across many server instances becomes difficult. Thus, along came orchestration options like AWS Elastic Container Service (ECS) and Kubernetes, which enable DevOps teams to maintain the configuration and orchestrate the deployment and management of hundreds, if not thousands, of containers.

While there are many similarities between AWS Elastic Container Service (ECS) and Kubernetes, there are different terms used within each. For simplicity's sake—besides when talking about Kubernetes specifically—we refer to "containers" and "server instances" where Kubernetes would refer to "pods" and "nodes."

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. Shared resources like containers cause challenges with cost allocation, cost visibility, and resource optimization. In the containerized world, traditional FinOps cost allocation doesn't work. You can't simply allocate the cost of a resource to a tag or label, because resources may be running multiple containers, with 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 cloud server (e.g., EC2 instance, virtual machine).

Pod
: This is a Kubernetes concept. A pod consists of 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 ECS.

Cluster
: A group of server instances, managed by container orchestration.

Namespace
: Another Kubernetes concept, a namespace is a virtual cluster where pods/containers can be deployed separately from other namespaces.

A Kubernetes cluster (see Figure 18-1) 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.

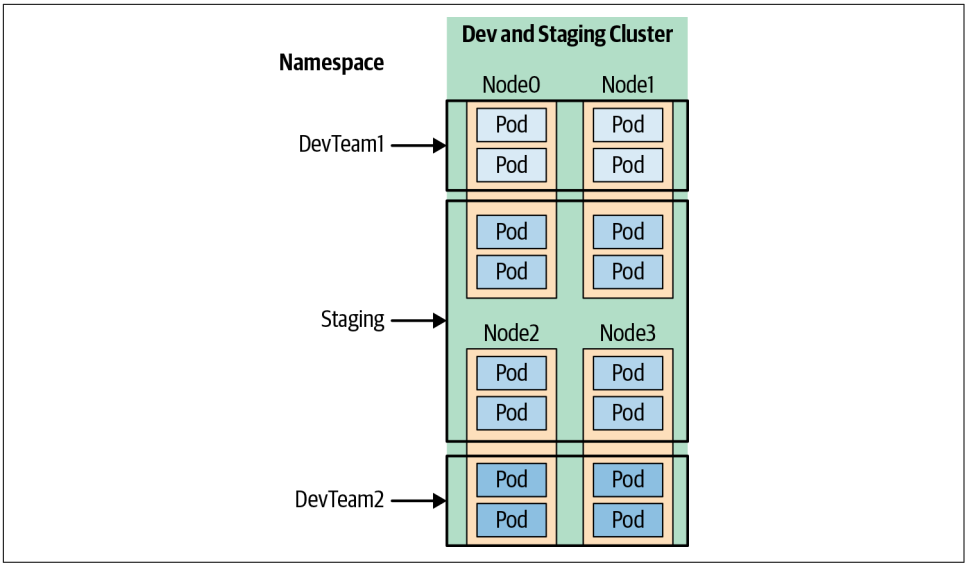


Figure 18-1. Basic view of a Kubernetes cluster

The Move to Container Orchestration

Remember, FinOps is about making sure that you are doing what's best for the business. If you compare the total cost of a running cluster to the cost of moving each container onto its own virtual machine instance—sized appropriate to the container needs—you can see the difference in cost. But we're not suggesting the only benefit of container orchestration is cost optimization. *Cost should not be the only driver for the migration to containers*. Being able to determine the cost of containerization versus traditional server-based deployments gives you a cost metric that will be useful when you are discussing the business benefits of containerization.

When you change from a large number of individual cloud server instances to a cluster orchestration solution like Kubernetes or ECS, 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 gain efficiency.

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 in "Container classes within Kubernetes" on page 232.

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 each 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, if not months. But research shows that most containers often run for short periods, less than 12 hours on average.[1] With different size 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 227.

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 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.

1 "8 Surprising Facts About Real Docker Adoption," Datadog, last modified June 2018, *https://oreil.ly/Nek9-*.

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. 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.

In order 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 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 in order 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 18-1 shows an example to help you understand this data.

In the table, there's a server instance with 4 vCPU and 8GB memory that costs $1 per hour. This server is part of a cluster and during a particular hour runs containers 1–4.

Table 18-1. 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.

As this is a complex area of container proportion calculations, we have not yet seen a standard approach. When starting your container cost allocation process, we

recommend you consider adopting a third-party FinOps platform that has a built-in method of measuring these proportions.

Even if you allocate the costs of your clusters in the previous manner, you can still end up with unallocated costs. It's unlikely your clusters will be 100% used all of the time, which leaves some proportion of the cluster costs unallocated. Having spent time with organizations that are successfully allocating their container costs, we've identified two main strategies for dealing with these unallocated costs:

Assigning the idle costs to a central team
: Since you need an allocation strategy to make sure that your cluster costs can be 100% allocated, some organizations take those idle 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 idle 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 idle costs.

Distributing the idle costs to the teams using the cluster
: The second strategy is to distribute the idle 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 idle costs.

  For example: team A runs a container that uses 20% of an instance, and team B runs a container that uses 40% of the same instance. That leaves 40% of the instance idle. Those costs are then allocated proportionately to team A and team B, based on their usage.

Before deciding on an approach for allocating the idle 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 GCP

When you use custom machine types, GCP 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 GCP 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 containers into a virtual cluster. Creating namespaces for teams, or for specific services, allows you to create a coarse-grained cost allocation group.

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.

Idle resources for containers

With clustered containers, it's true that underlying server instances are less likely to be idle, with multiple workloads running at the same time. However, there's a risk that individual containers scheduled onto your clusters will be inactive. As you now know, containers consume cluster resources, so if a cluster is larger than needed, it will be less cost-efficient.

Instead of tracking server utilization metrics looking for idle server instances, you're now required to track container utilization metrics in order to identify unused container instances. Again, the same processes that you use around idle server instances apply to containers.

Rightsizing clusters and containers

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 for rightsizing: horizontal rightsizing, vertical rightsizing, and workload matching.

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 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.

Container classes within Kubernetes

We mentioned earlier that cluster orchestration solutions like Kubernetes allow you to set different resource guarantees on the scheduled containers called Quality of Service (QoS) classes.

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.

Burstable resource allocation
: Spikey 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 (t-series from AWS and f1-series from GCP), which give you a base level of performance but allow the pod to burst when needed.

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 GCP or spot instances in AWS.

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 to 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 reserved, 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 13.

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.

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, in the serverless world, there are currently no reserved instances. That could mean that the cost of running all of your workloads on serverless might be significantly more than self-managed clusters. Having the cloud service provider manage the cluster also 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.

This brings you to the final chapter of the book, on unit economics, or what we like to call *FinOps nirvana*. This advanced FinOps technique encapsulates everything we've discussed so far.

CHAPTER 19

Managing to Unit Economics: FinOps Nirvana

Now that you've walked through the three phases of the FinOps lifecycle, you've arrived at the most exciting—and challenging—shift in thinking about cloud spend. The nirvana stage of FinOps is getting to unit economic spend management (what is our cost per X, where X = customer, rendering, user, etc.). It is important to get to that level of visibility so you can make informed decisions about when to spend more, or less, in cloud. And to do that you'll need to flex all of your FinOps muscles as well as bring in help from some other disciplines.

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've likely gone through the FinOps lifecycle a few times, refining your allocation strategies, metric thresholds, and 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. 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 through unit economics. Metrics help you identify the value in allowing a certain amount of spend to capture additional revenue, as well as set goals and key performance indicators (KPIs) on business performance.

Before the end of this chapter, we'll discuss how you're still missing some key business elements to achieve unit economics nirvana. To fill the gap on those broader

elements, you'll need to look beyond what FinOps can provide to technology business management models.

Metrics as the Foundation of Unit Economics

A common definition for *unit economics* is direct revenues and costs, associated with a particular business model, that are specifically expressed on a per-unit basis. Metrics enable you to determine the revenue you'll gain from a single unit of your business and the cost associated with servicing it, ultimately revealing the business value of spend. 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 a seat.

The concept of a single metric that measures the core value of a service isn't new. In fact, it's often called a *North Star metric* (NSM). Think of it as the guiding light that helps organizations focus on long-term growth instead of short-term influences.

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 a single offering served out of their cloud environment most likely need only a single value that's tied to their overall cloud spend. Organizations with complex structures and multiple product offerings will often have multiple units, each one relevant to a specific product line, application, or service.

Spending is fine, but wasting is not. Cast your mind back to the inform phase of the FinOps lifecycle. This phase emphasizes surfacing cloud costs to build awareness and ownership of costs. By having 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 metrics give context to cost optimizations within metric-driven cost optimization, a unit metric provides context to changes in cloud spend.

Coming back to the NSM, we often see organizations 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 entropy, 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 19-1, 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.

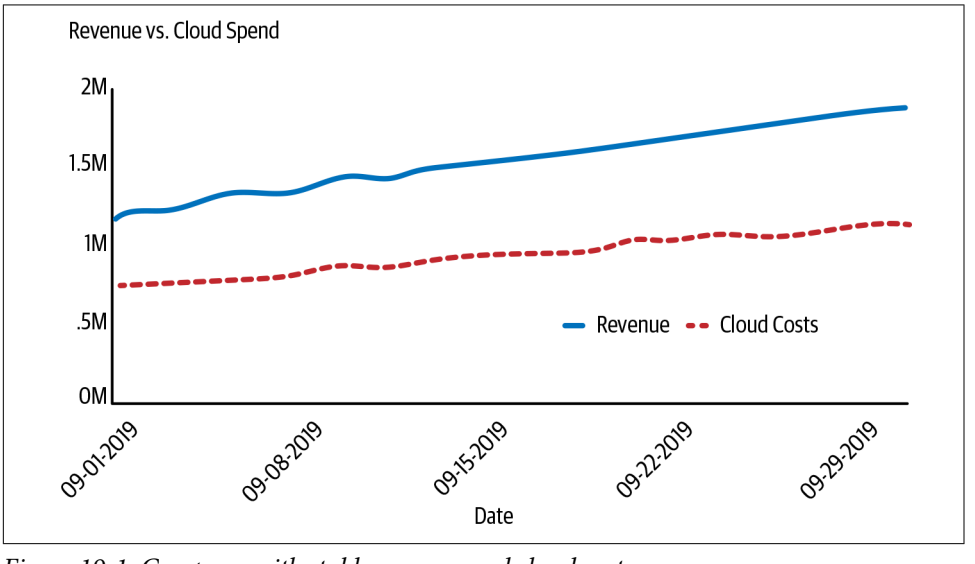


Figure 19-1. Company with stable revenue and cloud costs

In Figure 19-2, 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.

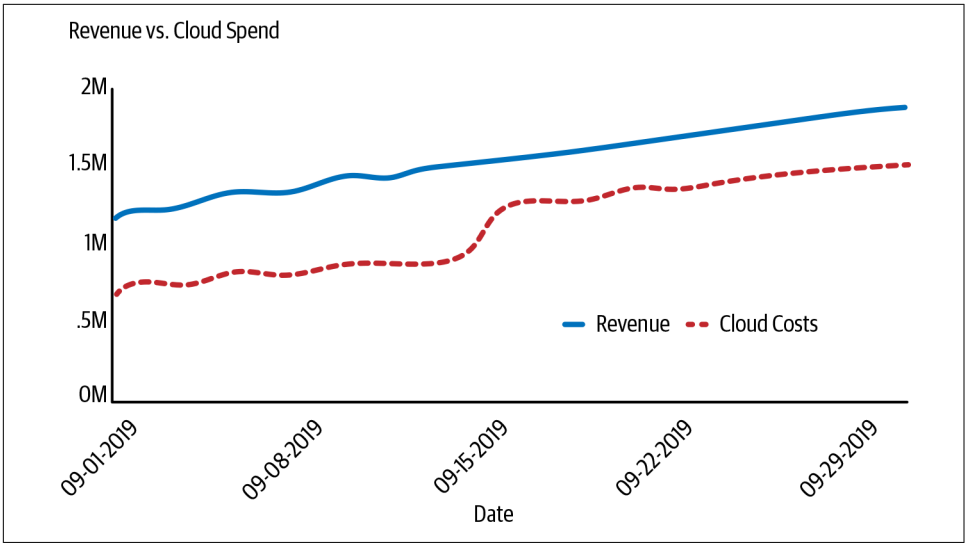


Figure 19-2. Company revenue and cloud costs during the introduction of a free-tier offering

In Figure 19-3, 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.

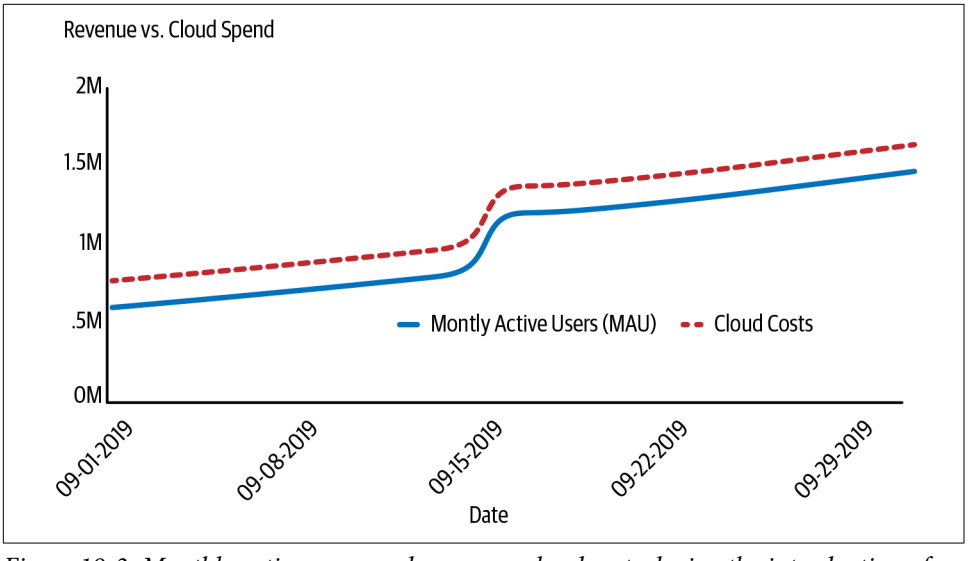


Figure 19-3. Monthly active users and company cloud costs during the introduction of a free-tier offering

Changing from corporate revenue to a unit like monthly active users, seats booked on flights, or transport movements tracked 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.

Coming Back to the Iron Triangle

Unit economics provides teams the data they need to measure—using the NSM—how an infrastructure change will impact costs. Conversations between operations teams and finance become less about the details of the change and more about the impact it will have. This allows the business to make decisions that weigh the benefits of the change against the expected impact to the unit cost, such as whether increasing the cost per active user by 2% would boost application performance by 10%.

When you apply unit economics to the optimize phase of the FinOps lifecycle, your goal setting matures from using only operational metrics, like hitting 80% RI coverage, to using revenue-impacting goals like reducing cost per subscriber by 25%. The former could help the business, but it could also be a distraction from what's really important.

Stories from the Cloud—J.R.

Let's look at an example that resulted in *increased* spend. In 2018, I worked with a SaaS company that came to realize if they spent more, they could improve customer experience significantly while protecting their revenue. A part of their system was having performance issues that caused slower load times, resulting in customers threatening to take their business elsewhere. Several engineers reported that they could dramatically shorten load times if they were allowed to increase the footprint of the application's infrastructure.

Luckily, this same engineering team was also considering cost as part of their process. They had diligently tagged their resources and could closely watch the costs associated with their deployment. The day after their initial test, they were able to calculate the expected monthly cost for the change and have a thorough conversation with the product owner about whether the additional $10,000 per month would be considered worth the added expense.

Together, they determined that the extra cost made business sense, and, with full approval from senior leadership, the change was put into production. The product owner was able to go to her finance counterpart and inform him about the budget variance before it happened. With no surprises that might undermine the process, the tech, business, and finance teams collectively built trust as they each agreed on the right choice for the business. That growing trust between teams helps accelerate the cultural change that is needed for FinOps adoption.

The challenge is to take efficient action while also maintaining growth and velocity. When you're in a highly competitive space, the goal is often less about cost savings than about driving more features and faster growth. Spending more in cloud might allow you to drive more revenue, can be a sign of growth in the customer base, or can enable you to more quickly shut down a data center.

Deciding to tighten the belt

Let's look again at the example of offering a free tier to your cloud offering. When measuring the increase in your cloud spend and the impact the free tier is having on long-term revenue, you determine that the increase in cloud spend is costing the organization more than the projected increase in revenue. You decide to reduce the cost of free-tier customers, possibly by using lower-performing resources for them. This reduction in free-tier performance might be all that's needed to tip the long-term benefits for a free program back into the business's favor.

Deciding to invest more

Alternatively, you could determine that the free program is working out well for the business, and projections show that further increasing application

performance or availability would lead to further increases in revenue. The business decides to increase cloud spend to drive up application performance. This pleases customers on the free offering, which in turn prompts more of them to convert to the paid tiers.

In either case, you make an intentional choice based on cross-functional conversations. Unit economics provides data to help you make decisions around the Iron Triangle.

Activity-Based Costing

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 (the cost per file created, cost per render, etc.).

In Figure 19-4, 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.

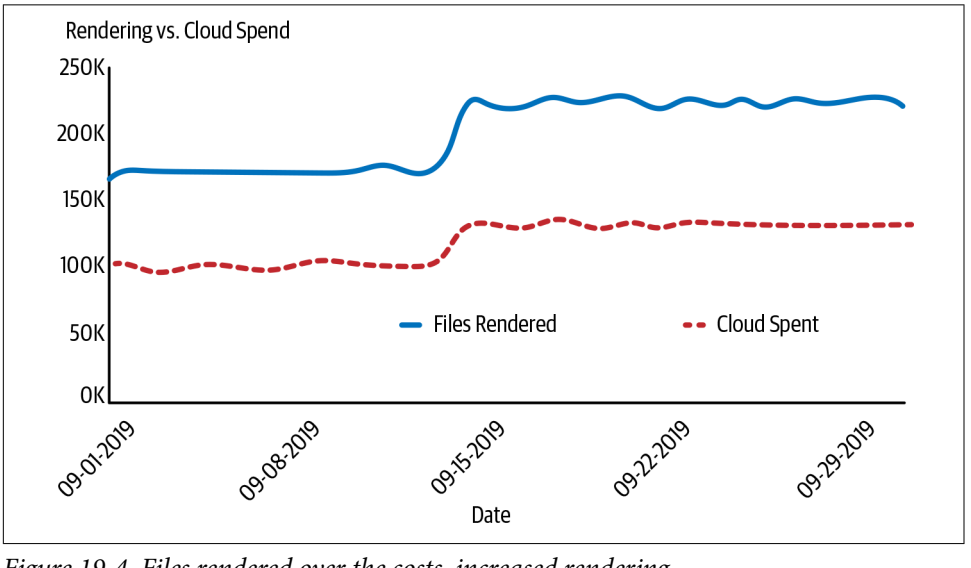


Figure 19-4. Files rendered over the costs, increased rendering

In Figure 19-5, 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.

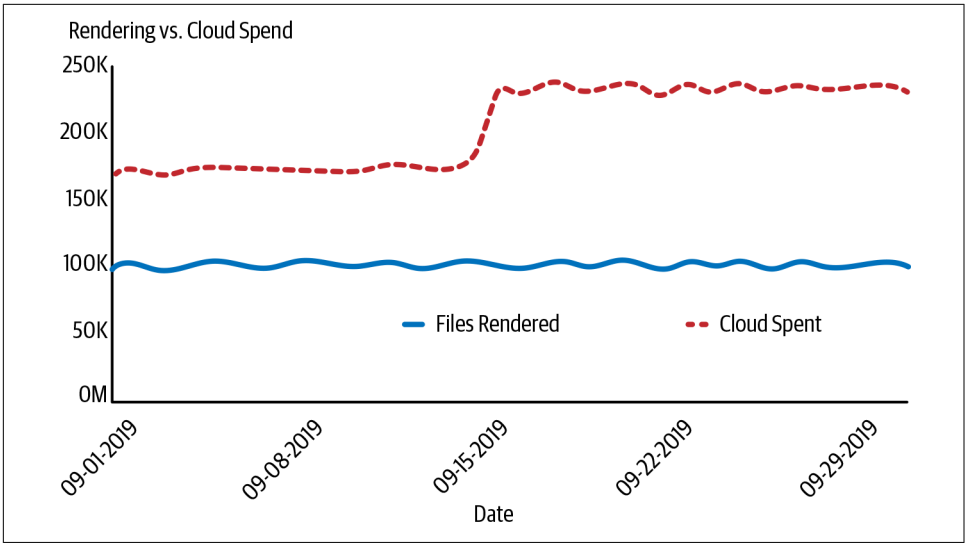


Figure 19-5. Files rendered over the costs, increased cost

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 RI/CUD portfolio. When you're doing activity-based costing, you're actively managing cost drivers.

What's Missing from the Equation?

So far, you've been using cloud spend as the numerator; however, you can incorporate a number of other costs. You've reached FinOps nirvana, and this has given you insights into the business benefit of your cloud spend, but cloud costs are only one part of the overall picture. After all, infrastructure costs often pale in comparison to labor costs.[1] Unit economics 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 only the cloud spend 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 11, we discussed refactoring services into a serverless architectural model. When deciding to rearchitect services, you should weigh the infrastructure charges

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*.

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 course, you could extend FinOps to include costs from outside the cloud bill. However, in our opinion, it makes better sense for FinOps to work alongside existing, tried-and-tested total cost management models.

Technology Business Management (TBM) has provided a framework for associating value with IT spending for years. However, TBM doesn't help organizations operate in the per-second, variable-charge cloud world, especially with the buying decisions distributed around the organization. This is why we believe FinOps is a great companion to TBM.

All too often we've sat down with companies in which the people running the infrastructure have no visibility into the revenue being generated by it—especially in tech. Conversely, those responsible for the revenue don't have enough visibility into the drivers of the cost behind that revenue. TBM brings revenue into the equation and marries it to the cost drivers that FinOps delivers, enabling a more complete business conversation about the revenue impact of cloud than either discipline can provide on its own.

We believe this is the line where FinOps should stop and then plug into existing well-established frameworks to inform decisions about IT spend. We have yet to see many success stories from those who marry these frameworks together, but we propose that FinOps will enable TBM to operate in a more cloud-native way, as well as within the cloud world, while TBM will allow FinOps to reason with the greater business costs outside of the cloud bill without reinventing the wheel.

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:

- Your North Star metric measures the business value of your cloud spend. Monthly active users (MAU) is a common NSM, but it will vary from business to business.
- 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.

As we close out the book, we want to take a moment to cast our minds forward.

What's Next?

The need for FinOps all started with cloud becoming an integral part of organizations, which led to issues managing cloud costs. FinOps is an evolving art to fully integrate cloud costs into the business, and maturing your FinOps practice with your business will help you achieve true cloud success.

We want to highlight that there is no "done" when it comes to FinOps. We can't offer you the line 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. Some services and methodologies get deprecated, while new ones move in to fill the space. In short, as long as cloud is evolving and changing, so will FinOps.

Unit economics within FinOps is the point where we're seeing the most successful practitioners operate today. Looking forward over the next 12–18 months, we believe FinOps will progress with a connectivity into other larger business frameworks, such as TBM, to truly quantify cloud spend. For this to work, the frameworks will need to iterate and become more aware of how services operate today—especially in the cloud. With a "you build it, you run it" operational model, more teams around the organization contribute to services offered by the business. They're responsible for more individual microservices and often share costs between multiple business products. The way costs are measured outside of the cloud bill needs to be updated to understand the adoption of microservices, DevOps, and public cloud.

We started the FinOps journey with the goal of understanding cloud costs and making informed business decisions. By connecting FinOps to frameworks like TBM, you resolve the inability of these frameworks to understand cloud, but in return FinOps gains access to the total overall costs and revenues of the greater business. Working together, the systems make it possible for seamless integration of cloud costs into the greater business systems.

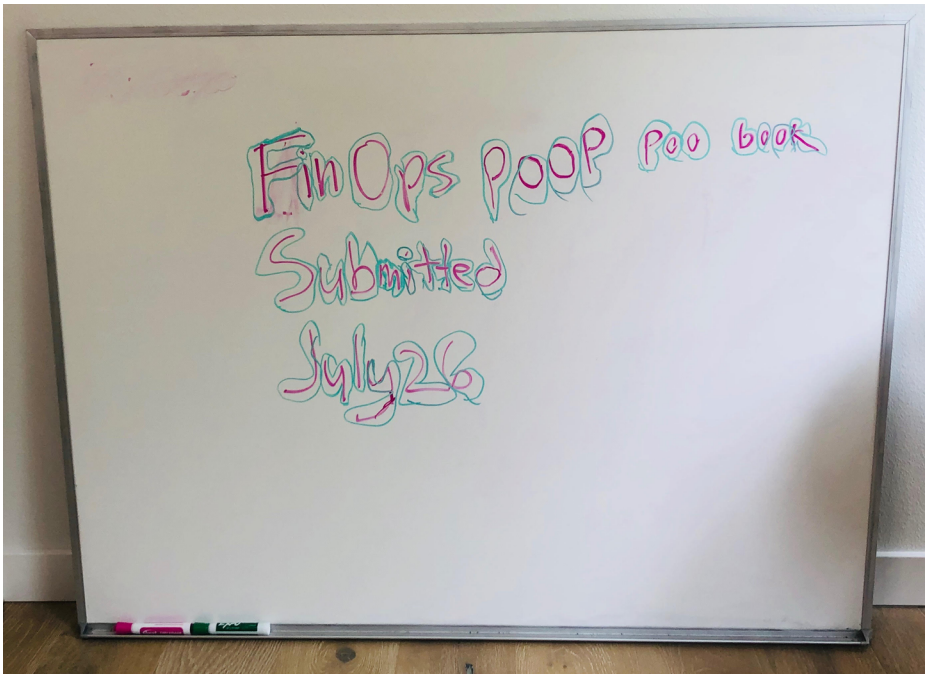
And by doing so, they make cloud success truly possible.

Afterword on What to Prioritize (from J.R.)

Ten days after the deadline for the submission of the first full draft 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 26th, 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 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: *https://bit.ly/2njjJAm*.

While this book is here to help your FinOps practice, don't let an expiring RI or rightsizing sprint keep you from going home to be fully present with your family on Friday evening. Even if that RI 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

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

About the Authors

J.R. Storment is the cofounder of Cloudability (now Apptio Cloudability). He's spent most of the last decade with hundreds of the largest cloud consumers in the world—from GE to Spotify to BP to Nike to Uber—helping them design strategies to optimize and analyze their cloud expenditures through technology, culture, and process. He's now VP of FinOps at Apptio and President of the FinOps Foundation. J.R. has spoken on cloud financial management 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 and worked in San Francisco, Barcelona, and London and now resides back in Portland, Oregon, with his wife, Jessica, and son, Oliver. He is the father of eight-year-old twin boys, one of whom passed away during the writing of this book.

Mike Fuller has worked at Atlassian's Sydney, Australia, head office for the past seven years, currently as a principal systems engineer in the Cloud Engineering Team (Cloud Center of Excellence). Mike's role at Atlassian has him working with most of the AWS services and assisting teams inside Atlassian to operate with security, high availability, and cost efficiency. Atlassian's Cloud Engineering team is responsible for the design, governance, and implementation of best practices throughout Atlassian's massive cloud architecture.

Mike holds a bachelor's degree in computer science from the University of Wollongong and nine AWS certifications. He has presented at multiple AWS re:Invent and AWS Summit events on topics that include AWS security and cost optimization (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.