the duckbill group

The Unconventional Guide to AWS Cost Management

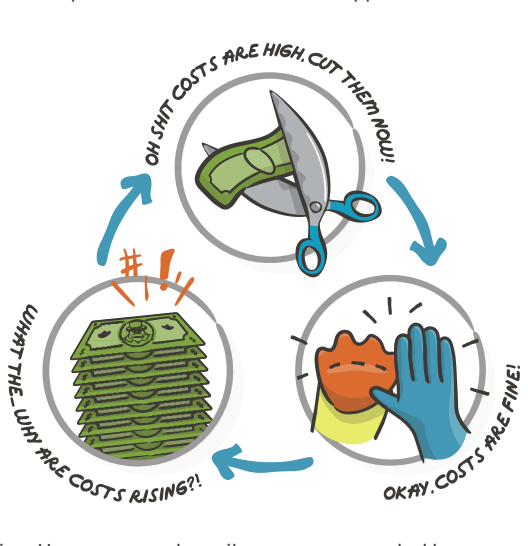
Do you find yourself struggling to answer Finance's questions about why the AWS bill is so high? Or the product manager's questions about what it costs to run their product? Did you just get handed a list of 50 AWS account numbers and a directive to "unravel this mess"?

If so, you're not alone.

Managing cloud costs is a problem every company faces as their cloud spend grows, and many vendors are seeing this as an important space to attempt to service. In fact, the market for cloud cost management tools is expected to grow at an estimated 20 percent annually through 2022. But these platforms are not a magic fix-all because the cloud spend problem doesn't have a quick solution. You can't easily solve it by throwing some software at it. And you can't easily solve it by tagging your instances and turning stuff off, either (though both of these solutions can help!).

In our experience, many organizations view cloud costs as isolated problems because the work happens at the sharp end, where engineers spend their time.

This misperception leaves organizations stuck in a constant cycle.



Managing your cloud costs in a way that creates lasting impact starts with proactively thinking about cost implications in all workflows—not just when costs get "too high." And when we say all workflows, we don't just mean Engineering's workflows. Building and maintaining cost-effective workflows should be a collaborative effort across multiple departments and functions within your organization.

For example, think back to your last quarterly planning session. How many roadmap items were requested by or required coordination with other teams? This could be anything from building new features based on the Product team's research to changing existing features based on the Customer Success team's customer feedback surveys to changing data models because an upstream dependency keeps breaking their API and all you can do is patch your own service to keep up.

Engineers may ultimately be responsible for deploying all these changes. But other teams—like Finance, Product, and leadership—all have a stake in this, too.

Suffice it to say that managing and optimizing your cloud costs is a complex problem. All of these interdependent moving parts need to be broken down into smaller, manageable chunks. And those chunks need to be aligned on the roadmap in order to ensure the work actually achieves your desired business goals.

Through our work with clients, we've found the companies managing their costs most effectively share certain traits and perform specific, regular activities. To that end, we've developed a model to make it easier for any organization to achieve success with reining in their cloud costs.

How Duckbill's Advice Is Unique

Simply put, The Duckbill Group believes that the main driver of costs stems from your application's architecture, so our approach views AWS spend through a lens that takes those unique circumstances into consideration.

Most articles opining about cost management tend to talk in vague generalities and only on the obvious-but-not-impactful topics, since really solving the problem takes much more attention and effort than running a quick script or buying a SaaS cost management tool.

Using the model we've developed allows us to quickly grasp where a company is at with respect to how well they're doing with cloud cost management, as well as to easily discuss the topics with other people who aren't spending every waking moment thinking about the problem space like Duckbill does.

Framing cost management within a model provides a baseline—a shared understanding for communication, delegation, and action that creates real, lasting change. A model ultimately allows all the people in your organization to constructively contribute to your end goal (i.e., keeping cloud costs under control).

For example, you've most certainly run into recommendations to turn off a bunch of idle resources. There's just one problem: those resources just happen to be your Disaster Recovery failover site. :facepalm:

Even worse, other approaches tend to treat your organization as a single, cohesive entity capable of moving in perfect lockstep. Sadly, we've never encountered this mythical organization. Our clients are all made of multiple teams with varying levels of feature velocity and interaction with each other. It's completely ineffective to treat all teams the same when it comes to cost management, as each team likely has different needs.

Additionally, existing models allow you to iterate in silos that focus on forward momentum, getting from point A to point B. But what if your team's path is different from another team's path? What if those paths depend on each other for forward momentum or—even worse—what if their end states conflict with each other?

Duckbill's approach provides a comprehensive framework with less specificity and directionality. Ultimately, this gives your organization greater flexibility in how to achieve your goals without reducing your ability to act on specific action items.

Our model contains four main capabilities:

Architect

Attribute

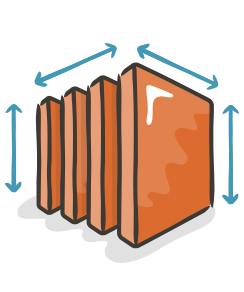
Invest

Predict

The Duckbill Cloud Cost Maturity Model

Each capability is a group of levers your organization can pull to influence its cloud costs. Capabilities relate to each other and interact with each other. But they are not contingent on each other for describing maturity. Such flexibility allows our model to flow with the uniqueness of your organization and provide a more representative description of an organization's maturity as it pertains to cost management.

Capability #1: Architect



An organization's main driver of cloud costs is not unused or over-provisioned resources, despite what seemingly every blog post out there likes to opine on. It's the application architecture. This capability covers the lifecycle of your data, data movement, waste reduction, cloud-nativeness, and ephemerality. Here, you will need to understand top-level services, fully managed services, and external services that integrate with AWS, such as Datadog and Databricks, if you use them.

Capability #2: Attribute

Understanding where your costs are going along business lines is crucial to being able to make well-informed decisions about engineering investments and cost optimization efforts, and to inform budgets and forecasts. This capability covers tagging and other attribution methods, at both a basic and advanced level, as well as the business systems created by attributing those costs.

Capability #3: Invest

Pre-purchasing resources and negotiating contracts is great when it comes to saving money on resources you're committed to using. Effective practices and processes around these investments are crucial for ensuring you take advantage of maximum discount levels. This capability covers Reserved Instances, Savings Plans, and contractual discounts through methods like the Enterprise Discount Program (EDP), Private Pricing Addendums (PPA), Migration Acceleration Program (MAP), and other less common methods.

Capability #4: Predict

Defining and tracking unit costs is important for understanding your infrastructure's cost drivers and the interplay between cloud costs and actual usage. Effective tracking of unit costs allows for accurate forecasting of AWS spend. This capability covers the identification of unit cost KPIs, forecasting capabilities, and maintaining your relationship with AWS to navigate the changing landscape of cloud before it impacts your business.

Interested in seeing where your organization stacks up on cloud cost maturity? **Talk to us about an assessment.**

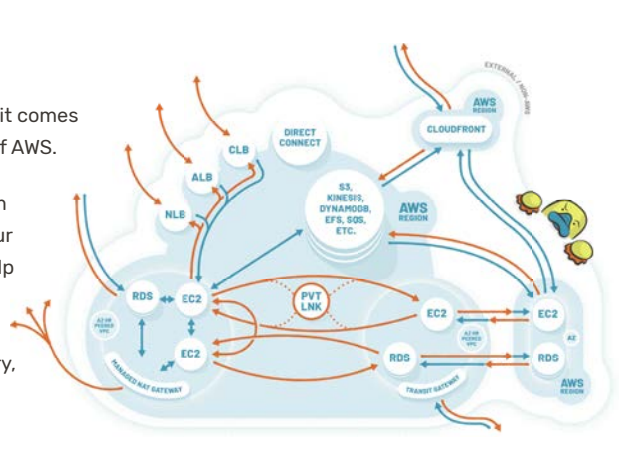
Contact us at **sales@duckbillgroup.com**

10 Tips for Optimizing AWS Cloud Spend

With that preamble out of the way, let's talk specifics. We've gathered 10 sticking points we commonly see that can help you start moving some of the levers within your organization immediately. These items are presented in no particular order; each is equally impactful.

This isn't your typical list of suggestions, though. These recommendations focus on unconventional nuanced work that will help to improve your architecture, attribute your costs more accurately, invest in both people and technology, and better predict your future forecasted spend.

Context Is Gold (or Silver)



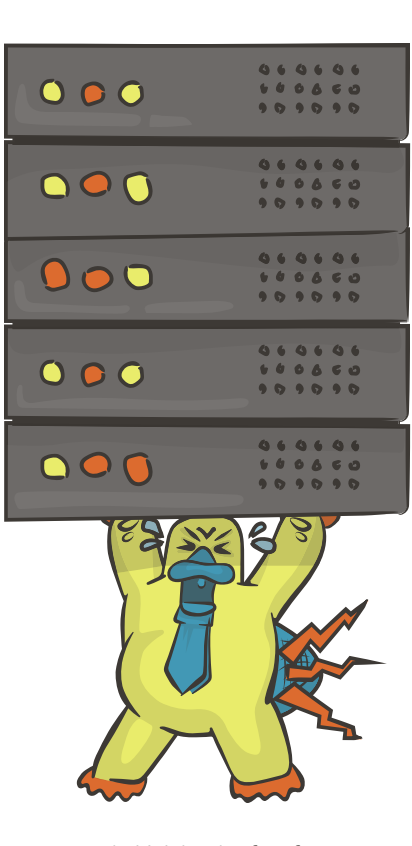
As we mentioned earlier, context is everything when it comes to understanding and improving the complex costs of AWS.

Remember that applications are greater than the sum of their parts. Understanding how components of your apps interact across the various AWS services will help you identify each app's communication paths. This will help you decide which paths are required for business objectives—and which ones are unnecessary, and can therefore be removed or rearchitected.

During an engagement with one of our clients, we found a large amount of data transfer flowing between Kinesis, S3, and Aurora. Most of this data transfer cost was hidden within the data IO costs of Kinesis and Aurora. But after chatting with the various siloed engineering teams, we learned that most of this data was simply duplicating streams for different teams to use. By migrating to Kinesis Enhanced Fanout, this client was able to dramatically cut their costs while simplifying their streaming data infrastructure. Without the context of what was going on, we never would have found such an opportunity.

One way we do this ourselves is by mapping data flow and dependencies on architectural diagrams and working out exactly what the client's business needs are—including needs around data retention, availability, and durability. This naturally will lead to discussions around cost for various components. Bonus points if you can overlay the costs onto the diagrams themselves! This will make it easier to identify costly infrastructure components and the context necessary to evaluate whether optimization work is needed.

The Cloud Is Not Your Data Center



If you're migrating from a data center to the cloud, make sure your roadmap includes plans for the lift-and-shift *and* the migration to cloud-native services afterwards. Treating the cloud as just another data center by only using the core services (e.g., EC2, EBS, S3) is hideously expensive. Contrary to what it might seem, the more you use AWS managed services, the cheaper AWS becomes. According to Scott Buchholz, a managing director at Deloitte, this pattern is a common scenario for IT departments: "They treat the cloud like a virtual data center and they don't change their operations or procedures when they move to the cloud."

If you treat the cloud like a data center, you might as well just have stayed in the data center to begin with and saved yourself a whole boatload of money from the migration. Not only are you missing out on the potential cost savings benefits from using some of the higher-order managed services like Lambda, Redshift, or DynamoDB, you are also missing out on all the other benefits they can provide. To illustrate:

Services like Lambda can give your engineering teams a streamlined way to quickly build, test, and deploy functions to your end users without having to think about traditional concepts like servers and scaling.

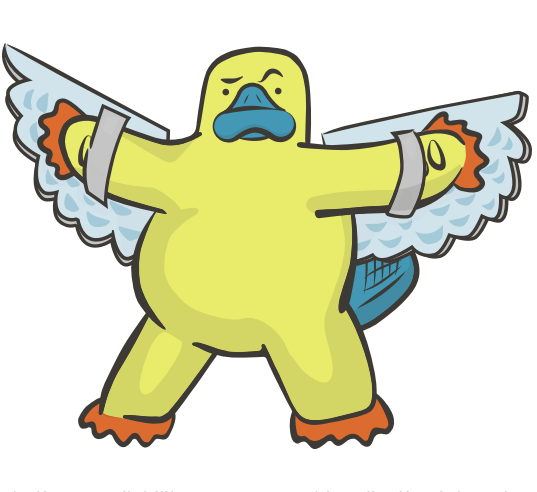
Using a backend database service like DynamoDB or Aurora gives your teams access to high levels of performance and availability—far above what could be designed and built on top of services like EC2.

Most importantly, these services require significantly lower levels of ongoing maintenance to ensure their ongoing operation. But, since we are talking about cost savings, don't forget the opportunity cost of running your own database servers. Think about all the various engineering efforts you can devote your technical teams to when they are no longer needed for ongoing database maintenance.

One area where this is particularly prevalent is with migrations to the cloud. While lift-and-shift is a great strategy for migrating workloads into the cloud quickly, this approach puts you on a shot clock. The longer you hold on to your legacy infrastructure in the cloud, the harder it will be to complete the migration and actually achieve all the performance and cost benefits that the cloud provides.

More importantly, you'll be racking up technical debt as time passes from the original lift-and-shift strategy. This technical debt can slow down your future product roadmap plans. And it can also wreak havoc on the productivity and effectiveness of your engineers who need to spend time to maintain this fragile infrastructure. The added risk you will begin to see is the loss of these engineers as their frustrations grow over time. This "brain-drain" will create a cycle that will make it much harder to modernize and migrate your workloads.

Moving Data Is Expensive & Painful (Just Like Moving Banks)



When it comes to data in the cloud, everyone is quick to call out their growing data storage costs including everything from S3 to EBS volumes. But many people forget one of the largest sources of data costs: the movement of that data.

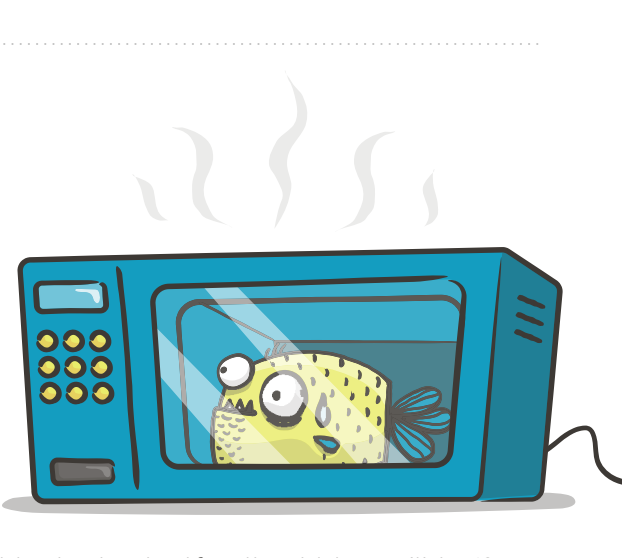
Moving data around in AWS is expensive. Work with your teams to understand your data movement patterns and business requirements and identify which patterns are necessary to meet your business requirements.

For example, do you absolutely need to replicate your data into three availability zones or would replicating into only two availability zones meet your needs (while cutting your costs)? Are you seeing a large increase in your S3 data transfer costs? Don't forget that you can use CloudFront to improve the user experience while dramatically reducing the costs associated with accessing data within S3.

The cost of moving your data around is more than just the "Data Transfer" line item on your AWS bill. For example, moving data around unnecessarily can increase the cost of your Aurora bill with all the additional charges for IO reads and writes. And if you're not careful, moving data from your VPC to other AWS services could inadvertently increase your NAT Gateway costs (you did set up all those Gateway and Interface Endpoints, right?).

The key point here to remember is that not all data transfer is the same. Many AWS services have no added cost for transfering data into and out of Amazon S3. Your goal should be to persist data into Amazon S3 as soon as possible. Then, after that data is safe and sound, use the various AWS services to ETL your data (Extract, Transform, Load) into whatever format your application requires. Understanding how AWS charges you to move data from one destination to another is one of the cost optimization tricks that can help you run enormous data platforms on AWS for the lowest cost possible.

Infrastructure Code Smell (aka Who Microwaved the Fish?)



Developers often talk about code smell, which refers to characteristics in the code that might indicate a deeper problem. But what about infrastructure smell? That would be all of the infrastructure that doesn't quite serve its purpose as efficiently as it could.

Infrastructure smell isn't just the idle resources contributing to waste (although that happens also). It's all the infrastructure hacks and one-off implementations deployed throughout your AWS accounts.

It could also be resources that have been provisioned and later abandoned and forgotten, databases with low IO configured to use high IO volume types, or Fargate or VPC logs stored in CloudWatch with no retention policy. Think about all the places you're using a hammer and nail when a screw and screwdriver would be more effective.

Infrastructure smell can occur with any new AWS engineering initiative. Building out new products, features, or integrations deployed on AWS without considering usage and cost impacts can dramatically affect your cost margins.

Waste takes more forms than just unused resources, after all. **Here are just a few places to start sniffing:**

Make sure your logs always have data retention policies in place.

Identify the business requirements for data retention, and configure your log storage to only keep the most recent logs (e.g., 90 days). If you need to keep logs for a longer period of time, consider moving them to an infrequent-access S3 storage tier.

Avoid using CloudWatch as a queue for metrics and logs.

Start by understanding the flow of your data into and out of CloudWatch. We often see companies log to CloudWatch (at a high cost) when logging to S3 would solve the problem at a dramatically lower price point. Similarly, pulling metric data directly from CloudWatch can become wildly expensive. Instead, consider implementing an observability router such as Vector or Fluentbit to route metrics and logs to the best (and cheapest) destination. Be mindful of third-party observability solutions that integrate with CloudWatch, such as Datadog and SignalFx, as the CloudWatch request API can get expensive. We've seen situations where the CloudWatch bill that Datadog is causing is higher than the Datadog invoice!

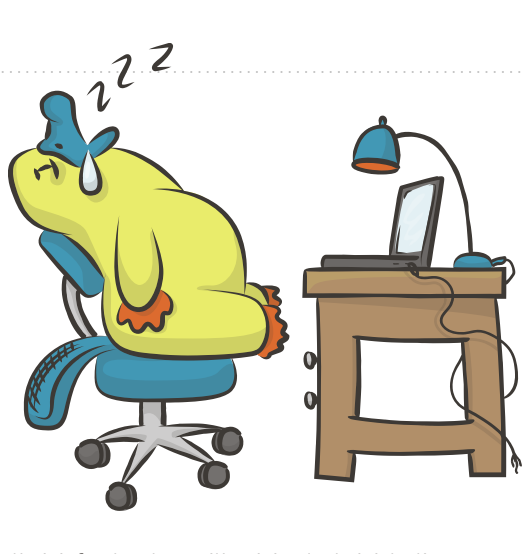
Optimize your compute usage.

AWS has an extremely handy built-in tool called AWS Compute Optimizer, which highlights the compute resources that are underutilized and overutilized based on the last 14 days of use. But be careful: Compute Optimizer only looks at CPU and network metrics by default. If you want Compute Optimizer to make recommendations based on memory utilization, you'll need to install the CloudWatch agent on your hosts.

Terminate unused storage — particularly EBS volumes.

EBS costs add up fast, especially if they're not general-purpose volumes. A quick CLI command can highlight EBS volumes that are no longer in use. You can implement an event-driven AWS native governance solution or you can leverage a third-party cloud governance tool like Cloud Custodian to provide custom reports of unattached volumes based on criteria like region, linked account, and user-defined cost allocation tags. Delete these volumes if they're no longer needed. If you want to save the data, take a snapshot of the volume before deleting.

Humans Are the Most Expensive Part of Cloud



Running your entire infrastructure on EC2 instances is a great way to start with AWS if you've never used it before. But it's also a great way to quickly rack up a truly horrific AWS bill without reaping the benefits AWS has to offer.

Instead, consider leveraging AWS's native service offerings. They're a less expensive way to operate in AWS than using raw compute resources.

AWS gives you a multitude of options to run your software on their infrastructure with minimal administrative overhead. Instead of deploying your software on EC2 instances, consider containerizing your software and deploying it to ECS—bonus points for Fargate on Spot instances—for a fraction of the price. Of course, you could also adopt a serverless architecture with Lambda, without any infrastructure to manage at all.

It's not just compute either. Need a new database but not sure how much storage it will need or how much demand there will be on it? Aurora Serverless provides database functionality with a dynamic storage volume that can grow or shrink with your data. When its workload is stable, you can migrate it to an Aurora RDS instance and take advantage of investments like RDS Reserved Instances.

But there are some places where it might seem harder to move away from AWS's basic compute resources, such as with stateful distributed systems (e.g., Kafka, Cassandra, and MongoDB). These systems require high compute, storage, and IO resources, lots of engineering hours for development and maintenance, and can easily generate a ton of data transfer charges if they're not designed efficiently.

Luckily, AWS has you covered here, too, with their native managed service offerings. On paper, AWS's managed service offerings may appear to be much more expensive than running the comparable offering yourself. But think about them from a total cost of ownership (TCO) perspective.

Any stateful, distributed system deployed on EC2 instances generally incurs several different buckets of charges: compute, storage, data replication charges, data query charges, plus the engineering effort to manage the entire system. Meanwhile, AWS's managed service versions of these solutions charge you a simplified price—and replication traffic is free, since they're running it all.

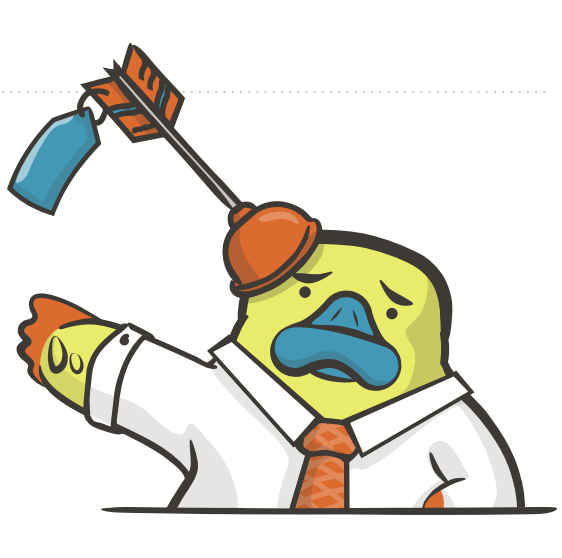
That's right: In most cases, data replication is free, and you don't need to put any engineering effort into managing the system long term. AWS handles that all for you.

Now, it's worth calling out that—in practice—determining the value of that administrative overhead is hard. For example, is your team already familiar with how to design and deploy a Kubernetes cluster? How much time are your engineers actually spending researching and building the infrastructure versus training other teams on how to deploy to it? Are there other business goals that your team could invest in rather than managing this infrastructure?

It's nuanced and situation-specific. But it should be involved in every architectural discussion. That way, you always know the tradeoffs you're making for your architecture. And, more importantly, you always know why you're making those tradeoffs.

Start asking these questions of your teams and departments to better understand your specific situation and organizational constraints.

Tag—You're It!



It seems like every blog post that talks about "how to save money" always tells us to tag our resources. But that advice feels much the same as when the doctor tells us to eat healthier.

Of course, we all know we should eat more veggies. But the doctor's advice doesn't teach us anything new about veggies. What we need to learn is how to make vegetables taste more delicious so that we'll choose to eat more of them.

In other words, our veggie consumption (er, lack of) isn't because we don't know that we should eat more. It's because we're never looking forward to a heaping plate of unseasoned lima beans.

Likewise, tagging your resources just for the sake of tagging won't help your organization because you're missing the crucial component: strategy.

For the best results, start your tagging efforts as a series of conversations with different members of the team. Invite the people who can have that conversation in a holistic way. Bring together Finance, Product, and Engineering/Operations to talk about how each team can leverage tagging to benefit their work. Collaborate to create a purposeful tagging strategy— one that will incentivize Engineering teams to commit to tagging because it will actually enable them to do their job more easily and more effectively. Track your tagging success (or lack thereof) at a similar granularity across teams or products. Celebrate the successes of teams that tag all their resources, thus creating accurate cost attribution, and socialize their workflows with other teams who may be struggling to tag or get value from their tags. Once you have a tagging strategy that everyone agrees on, you need to shout it from the rooftops.

Not sure where to start with a tagging strategy?

Lucky for you: AWS has a "thrilling" 24-page Tagging Best Practices Whitepaper. Don't let the length or the undoubtedly "edge of your seat" storyline sway you from giving it a quick perusal. It includes a slew of helpful use cases for various tagging strategies to fit nearly every business need.

Win Friends and Influence DevOps: Continual Tagging Improvement

Tagging is a continual process that requires buy-in from all people across the organization. The more you embed resource tagging into your organization's culture, the greater your long-term benefits will be.

Tags are for more than just billing or finding out "who spun up this x1e.32xlarge host?" (Trick question! The answer is always you. You just forgot to terminate it.)

Learn how different teams in your organization use tags and share that knowledge across the organization. Security might use tags for audits and patch compliance while Finance uses tags to identify cloud spend attributed to development work for tax credits. Both are completely valid use cases for different reasons and both generate different requirements for your organization's tagging policy. Socializing each team's tagging requirements provides business context for why including each tag is so important—which ultimately helps all teams reach their goals.

But all the socializing in the world may not actually help your teams to implement and use those tags. So, you may need to take a more targeted effort towards tag adoption.

There are two main strategies to enforce tagging within your organization.

The first strategy is to enable the continuous monitoring of your AWS resources and notify the appropriate teams and users when their tag usage falls out of compliance.

The second strategy, more commonly found in cloud-mature organizations, takes a hard-nose approach towards tag enforcement by terminating resources that do not meet the predefined requirements. This one requires significant buy-in from the organization because—while it may be fun to automatically terminate AWS noncompliant resources—pulling the rug out from your engineers may not win you many friends at the lunch table.

There is a case to be made that companies should adopt both of these strategies.

AWS Config Managed Rules can help to ensure that your resources are tagged according to your requirements by reporting on tag compliance and coverage. AWS Organizations can prevent the creation of resources that are missing required tags. While there isn't any AWS-built functionality for terminating non-compliant resources (yet!), the incredibly helpful open source tool Cloud Custodian works quite nicely to define policies and enforce compliance with those policies across all your various AWS accounts. Cloud Custodian can enforce compliance on much more than just tags, too.

No matter which strategy you choose to take with tagging your resources, know that the only way to ensure ongoing success is to have a plan for monitoring and compliance. Creating a process or even—*gasp*—automating your tagging remediation will help make sure that you are never falling behind on tag coverage.

Invest in Your Future

If you have highly static workloads throughout the day, invest in those resources and AWS will give you significant discounts on their usage spend. You already know planning Reserved Instance purchases can get...complicated. If you're not yet familiar with them, Savings Plans has changed the game for compute discounts by offering the same savings as Reserved Instances but without all of the management overhead.

But wait, there's more: Savings Plans apply to Lambda and Fargate—unlike Reserved Instances. The more aggressive you can be with your Savings Plan commitment—such as upfront cash commitments and longer terms—the greater the discounts you can achieve.

Sadly, Savings Plans aren't available for non-compute resources such as Redshift or RDS. But planning Reserved Instances there is much cheaper. If you're in doubt about whether Reserved Instances make sense for a particular workload, know that the typical break-even point is only seven months. In other words, if you believe the resource will be there for at least seven months, you'll save money by purchasing Reserved Instances.

And don't forget about all the other AWS services that support Reserved Instances, like Amazon Elasticsearch, Elasticache, and DynamoDB. There are even reservation discounts for some of the more obscure services, such as Elemental MediaLive and MediaConvert!

The key point here is to track your use of AWS services as they grow and stabilize over time. Make these usage commitments as soon as you have confidence in your baseline consumption to maximize your savings.

Why Are You Still Paying Retail Prices?

A little-known fact with AWS is that every account has an Account Manager, even if you've never met them. Trust us: They would LOVE to help you save money by investing in the AWS relationship.

Seriously, we aren't making a funny: While many people see their Account Manager as just the sales rep bent on making you spend more money, the truth is that they want to make sure you're happy with how much you're spending on their services.

They know that a customer who feels like they are wasting money on AWS is not going to be a happy customer. Remember, Amazon is in this for the long game—not short-term wins at the customer's expense.

Generally speaking, you need to spend about $1 million per year with AWS before you can take advantage of AWS's Enterprise Discount Program. Once you start spending more than that, you can sometimes be eligible for further custom pricing contracts on specific services, such as S3, CloudFront, and even data transfer.

Additionally, money spent purchasing Partial Upfront or All Upfront Reserved Instances and Savings Plans can contribute to your EDP commit requirements. Here's how it breaks down: When you purchase Reserved Instances or Savings Plans, you have the option to purchase with some form of upfront commitment. You can think of that commitment like a downpayment: The more you pay upfront, the greater discount you'll receive. Meanwhile, when you engage in a custom pricing agreement, AWS requires you to spend some amount of money over the course of each year of the agreement. If the services covered by your PPA or EDP offer Reserved Instances and Savings Plans, the upfront money you spend on those investments can contribute to your overall PPA or EDP commitment requirements for the year.

Remember, these are only available for companies spending at least $1 million per year. If that's you, talk to your Account Manager for more details.

Want help negotiating your AWS contracts? **Let's chat.**

Contact us at **sales@duckbillgroup.com**

Predict Your Future (and Make Your CFO Happy)

Imagine your company gains publicity and suddenly becomes insanely profitable. Sales is closing deals left and right. Customer Success is onboarding new customers around the clock. And your application is seeing an influx of user traffic.

How would your AWS spend change? How would your non-AWS infrastructure-related spend change (think PagerDuty, Datadog, etc.)?

Start by thinking about your architecture on a basic level. Perhaps your application is heavily compute-based, so you expect EC2 costs to spike as lots of new instances spin up to handle traffic. Or perhaps your application is heavily storage-based, so you expect S3 spend to go up as users generate and store tons of new data.

This is a great first step. But it only gives you a ballpark idea of cost changes, and Finance will probably want something a little more concrete and accurate.

Ideally, you want to know which components of your application are directly impacted by additional activity and how AWS usage for those components will fluctuate accordingly. To answer this question in a financially accurate way, your teams need to know your product's unit economics.

Unit economics describes your product in terms of revenues and costs in relation to a KPI that tracks closely with customer demand. That KPI can be any basic, quantifiable item that creates value for your product. For example, in the airline industry, the unit economic KPI is usually a seat on an airplane. If you sell a software agent-based security tool, you might want to figure out the cost to service each deployed software agent. If you operate more in the business-to-consumer (B2C) market, you may want to focus on your total costs per daily active users (DAUs) and track cost per 1,000 users. Or maybe your B2C products require factoring in multiple costs like data storage, data transfer bandwidth, and other charges incurred by each customer, regardless of how many of that customer's users are actually using your product. Some clients we've worked with even build different models for different application environments, prioritizing KPIs for production-related spend (normally referred to as Cost of Good Sold or COGS) over non-production-related spend (e.g., development, staging, and analytics). Your KPI and how you break out your costs to support your customers will be unique to your overall business.

Once you've identified your product's unit economic KPI, think about the infrastructure costs associated with it. These infrastructure costs include AWS and any other services you leverage to run the product, like Datadog, PagerDuty, Twilio, and Splunk. One simplistic approach to calculate your unit economic cost is to take the sum of all the infrastructure costs (including applicable third-party services) and divide that by your product's specific unit metric.

That process works great for an organization with a single product and a single set of associated cloud costs. But let's be real: You're likely running multiple products across dozens or hundreds of AWS accounts.

How do you accurately account for each product's total infrastructure spend when all the cloud usage is shared?

This is where a robust tagging strategy with high coverage comes into play. We mentioned the importance of tagging and socializing your tag governance earlier specifically for this reason.

Tagging isn't just about checking a compliance box. It's about setting your teams up for success by making data-driven decisions with accurate financial models and unit economics. The more your AWS spend is attributed to your products, the more accurate your unit economics will be.

We like to think of this as a cost error margin. If only 60% of your taggable AWS spend is attributed to your product(s), your unit economic data has a 40% margin for error. That's pretty high.

Fortunately, bringing that cost error margin down is an easily measurable goal. Start by measuring your current cost error margin based on how much taggable AWS spend is actually tagged. Then, set small goals to work towards decreasing that error margin down to the industry gold standard of 10% to 15%—e.g., 85% to 90% of your taggable AWS spend is attributed to your product(s).

Identifying and building your organization's unit economic model takes work. But the benefits are enormous. Once you know what a user costs, predicting your AWS spend is only a matter of estimating user growth, which allows Finance to predict more than just a quarter or two out.

There's also another less obvious benefit: Once you know the cost of a user and the drivers that lead to that, you can begin tweaking things to improve your margins. Imagine winning deals with competitive but profitable discounting because you have the confidence to know exactly what it will cost to service that customer. That's how you can leverage unit economics to grow your business in an increasingly competitive economic environment. Not only will you make your CFO happy with your ability to forecast the complex cloud spend, your Head of Sales will become your new best friend because you've given them a powerful tool for helping them price deals quicker and more easily.

I thought this was supposed to be easy!?

failed to implement their planned digital transformation initiatives.

We get it. You scrolled through this hoping to find 10 quick tips to click a few boxes in the AWS console and cut your bill.

Unfortunately, lasting change is not that simple. According to a recent survey of 328 large enterprises, 78% failed to implement their planned digital transformation initiatives.

Their biggest blockers? It wasn't technology.

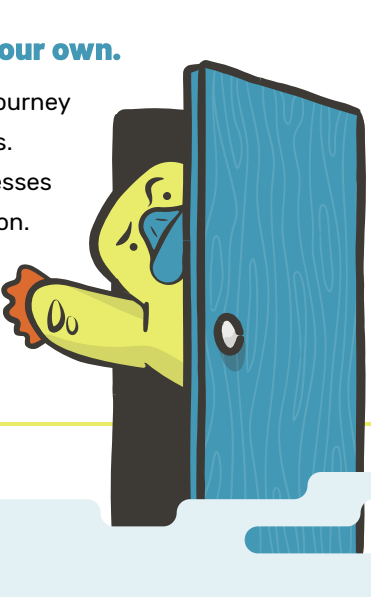
It was people.

According to the survey, four of the top five reasons organizations fail digital transformation initiatives are related to people: particularly organizational structure, governance, culture, and communication channels. But this doesn't just apply to large enterprises or technology transformation stories; smaller, cloud-native companies can learn from this as well.

Any organization leveraging the cloud needs to foster a culture that supports cost thinking. You can direct teams to complete individual optimization action items. But without a full company cultural shift to proactively think about cost in every business and architectural decision, you're going to end up right back where you started.

For a lot of you who've been around the block a few times, this might sound familiar. Today's conversation about the importance of culture in cloud cost optimization and management is the same conversation that organizations had about the importance of culture in DevOps. Building organizational commitment around cost savings and cloud cost management is a similar cultural shift. Bringing together stakeholders across Finance, Product, Engineering, and senior leadership on this shared mission takes a focused effort over many months and years.

The good news is you don't have to go it on your own.



If you want help from experts when you're starting your journey towards cloud spend optimization, check out our services. We have helped many teams build the foundational processes necessary to create lasting change across the organization. These companies have been able to both save money and predict, with growing accuracy, how their spend will change in response to evolving business demands.

Contact us at **sales@duckbillgroup.com**