Cost Management Strategies for Kubernetes

By: Bhargav Brahmbhatt . Technical Product Marketing Manager

Table of Contents

Introduction

The advent of cloud computing has enabled us to develop faster and ship more than ever before, but we're spending a lot more on cloud than we anticipated. And with Kubernetes a mainstay, managing the manager of your infrastructure usage is becoming a crucial leg of cloud cost management. At Harness, we've had the opportunity to see how hundreds of organizations implement cost management strategies. Consistently, organizations struggle with how to get deep visibility into their costs, how to optimize their infrastructure, and how to do effective budgeting and capacity planning.

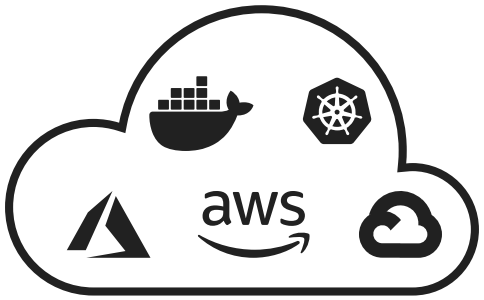
In this eBook, we take those learnings and share with you the most effective cost management strategies for Kubernetes.

Why Manage and Optimize Costs?

The answer is simple: it's the same reason you watch your own money. You don't want to throw away money, and you want to get the most value that you can out of the dollars that you spend.

Businesses operate by this simple principle, and want to keep costs as low as possible. Cloud infrastructure is a necessity for the business to operate, but that doesn't mean any amount of money can be thrown at it. When you buy food, it's because it's a necessity. But would you want to spend $1.35 on a banana when there's one next to it for $1, all things equal? Strange grocery store analogies aside, you don't want to spend an extra 35% on something when you know you don't need to. Cloud costs are the same, and yet, on average, 35% of cloud spend is entirely waste.

The bright side is that just as you can have a $1.00 banana, with a little visibility and knowing what savings look like, you can not only curb unwanted spending, but also extend to being able to project what all of your bananas will cost over time.



"35% of Cloud Spend is waste."

Source: AWS Reinvent 2019

Cloud Costs vs. Kubernetes Costs

But how does this relate to Kubernetes? The costs of running Kubernetes are a subset of your overall cloud spend, but also a significant driver of that spend. As an open-source technology, Kubernetes is actually completely free to use. The costs that come with Kubernetes are in the cloud resources that are consumed while Kubernetes is managing your infrastructure, such as spinning up compute instances. Kubernetes makes it easy to use and manage more cloud resources, driving up cloud spend at an accelerated rate.

Just as there are cost management strategies for cloud resources themselves, there are similar strategies you can use to manage and optimize your Kubernetes costs. In the end, you'll see that cost management for Kubernetes utilizes many of the same methodologies, and how to apply that same thinking to your Kubernetes costs.

Where Does the Problem Come From?

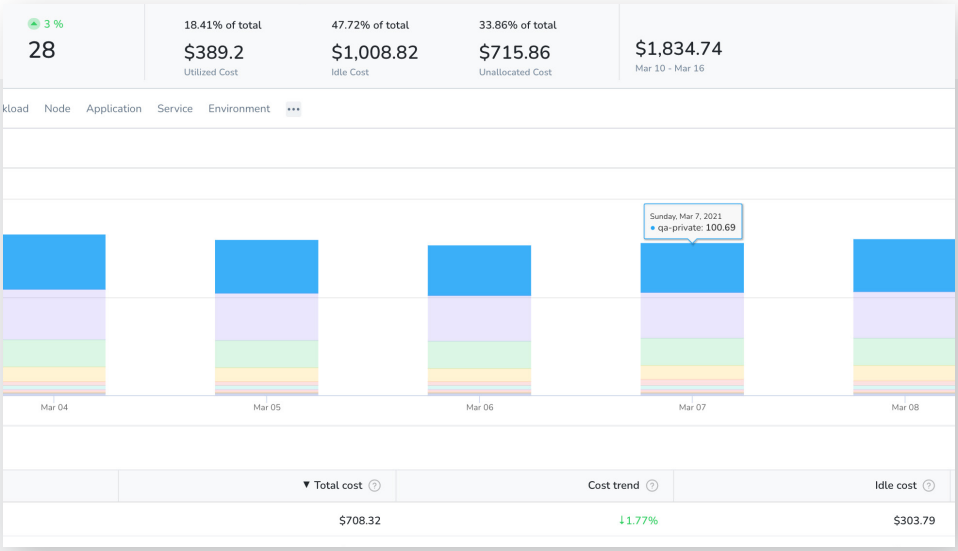
Here's the first way that cloud and Kubernetes costs are very similar: the core of it is the rise of the DevOps paradigm. In no way is DevOps a bad thing – in fact, it's the reason you're able to move so quickly and deliver innovation to customers faster than ever before. As part of this efficiency in software delivery, the use of Kubernetes in development and deployment is here to stay, because it simplifies the deployment, management, and scaling of applications.

A lot of the power of Kubernetes is in being able to take a defined need and freely scale up and down resource requirements. That need is defined by developers, who tend to overestimate resource needs so that things don't break. Intuitively, developers (along with DevOps and Operations alike) aim to maximize the core metric of application performance (which incidentally led to the rise of APM and observability tools). They err on the side of keeping apps running during the peak traffic spikes, but that costs a lot of money downstream. Thus, cloud cost management - and Kubernetes cost management, as a byproduct - were born.

However, it's important to realize that developers aren't the blanket problem here. In fact, you want to empower developers to freely innovate and build, test, and deploy code quickly. At the same time, you have to recognize that developers are the master keyholders to implementing cost efficiency, just the way that they are for software delivery and performance. Ergo, the key is empowering developers to be cost-efficient without added toil, the way you do for delivery and performance, is it not?

Crucially, cost efficiency is a function of more than just development and ops teams. To be truly cost efficient, it's just as important to empower developers and ops teams as it is to bring that empowerment to multiple levels of the organization, including engineering teams, ops teams, finance teams, and executives. In this way, you're creating and enabling a collaborative cloud cost management culture that brings the whole organization to joint goals, visibility into costs, opportunities to be more efficient, and effective capacity planning ability.

At Harness, we believe in this collaborative approach to cloud cost management, starting with the engineers. Providing this visibility, we've found, is one of the biggest roadblocks for engineers to create efficiencies in cost. Just like observability tools provide the information needed to improve performance, Harness provides what engineers need to identify and act on key opportunities to minimize costs.



Kubernetes Cost Management Strategies

To come up with the right strategy, it's important to first recognize the core goals that you want to achieve. Just as with cloud cost management, there are three goals to solve for to do comprehensive cost management in Kubernetes:

- Cost Visibility
- Cost Savings
- Cost Forecasting

The overall strategy is simple: first, see the costs; second, find savings; finally, achieve predictability in costs. Incidentally, these three steps are the same goals that will serve as the pillar for your Kubernetes cost management strategy, so you can achieve both at once.

What will follow are some strategies that can be taken within each of cost visibility, cost savings, and cost forecasting, so that you can create a comprehensive strategy depending on your organization's needs and maturity.

Cost Visibility

With cost visibility, you want to be able to reasonably attribute costs to their origins. Depending on the context of the business, this could be down to the level of the developer, project, application, service, business unit, or anything else that makes sense for your organization's needs. In Kubernetes, this can be achieved by looking at clusters, nodes, namespaces, workloads, and pods.

The most basic of ways to get visibility is using the Kubernetes API to see how resources are allocated, and then connecting them to a tool like Prometheus, to recognize what's being actively utilized versus unallocated or idle. This can work as a start but is difficult to scale, which is why you want to use a more holistic approach.

Let's dive into how you can achieve cost visibility to any level of granularity at any scale. These strategies are not mutually exclusive, and in fact, each provides an increasingly complete picture of cost.

Examples of What You'll See

With cost visibility, you'll be able to identify the contributors to your costs. In terms of contributions to wasteful spending in Kubernetes, you can expect to find:

- Cluster sizes bigger than necessary, even with Cluster Autoscaler enabled.
- Unallocated resources within a cluster that contribute to underutilized resources.
- Low pod density, indicating that resources are allocated to the cluster or node but not being used by any pod or workload.
- Mismatches between requests and actual usage of CPU and memory resources, resulting in overprovisioned pods or pod throttling.
- Shared storage across resources that may not be fully utilized, or storage not allocated to any pod that ends up as unallocated storage volumes you're still paying for.

Cost Visibility (Cont...)

Strategy One:
Tag and Label Management

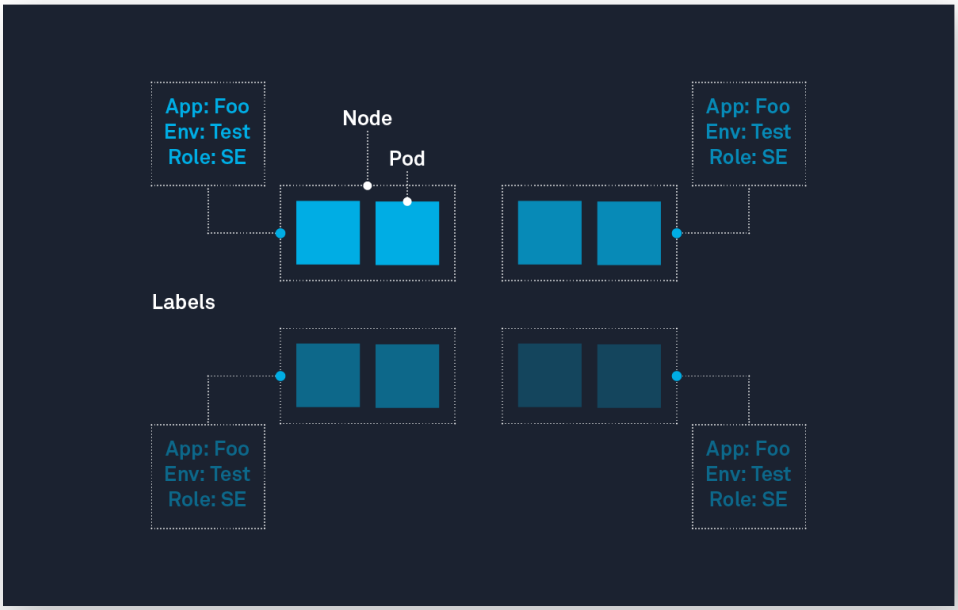
Tags are a native capability built into any cloud resource, and they're called labels in Kubernetes. They allow you to use a key-value style of assigning identifiers, or tags, to any resource so that you can find it later based on a combination of these pairs. It's great in that it provides you the full flexibility to create any kind of identification scheme, and you can tag resources manually or in code, making it simple to create good governance around resource identification.

With a robust set of tags and good tag management policies in place, it becomes possible to slice and dice the cost visibility and ownership into a relevant context for anyone who needs it. In an ideal world, you'd have all of the tags on every resource that everyone needs, to see costs in whatever way makes most sense for them.

However, because of the manual nature of managing tags as a core capability, it can become quite the challenge to scale this up. Tag management is hard to do, so tools which help here are valuable. You'll want to look for the following in a robust tag management tool:

- Ability to enforce OPA-based governance rules.
- Reporting that can surface non-compliant resources (e.g. showing what percentage of resource don't have the "Team" tag).
- Helps with tag compliance through automated tagging (e.g. add tags as part of a CI/CD pipeline, enabling you to answer questions such as "do frequent deployments impact cost efficiency?").

There are tools out there that make auto-tagging easier, and if you use those, you'll want to make sure you get full coverage of your architecture in any tagging effort. You'll also want to consider how you'll actually do the reporting if you don't have a visualization tool, dashboard, or some other way to collate tag information for chargeback, showback, or financial reporting.



Cost Visibility (Cont...)

Strategy One:
Tag and Label Management (Cont...)

Risks

Using tag management as your primary visibility mechanism into your infrastructure relies heavily on the teams using the resources to appropriately tag them. You have to ensure there are guidelines or governance in place that guarantee all resources are tagged appropriately; if resources aren't tagged, your entire visibility game plan could fall apart.

In addition to making sure there are no untagged resources, you'll want to consider that there are also untaggable resources. Examples of these are shared resources. If you're trying to do chargeback or showback for a shared resource, how do you attribute costs across multiple consumers of a resource?

When It's Most Useful

Tag management is great for two kinds of organizations:

- Small organizations that don't have complex reporting needs, aren't worrying about optimizing costs, or don't have a big infrastructure footprint.
- Organizations that can create and enforce robust tag governance policies and are able to leverage the tag data to meet reporting needs.

Typically, organizations that can effectively leverage the tag management strategy are early SMBs and enterprises with complex financial controls and governance already in place in other parts of the organization.

Strategy Two:
Organizational Mapping

If chargebacks and showbacks are the most important thing, then you need to map each resource back to the unit of your organization that's consuming it, whether that's by business unit, product, application, microservice, cluster, or workload.

At Harness, we've regularly seen that this is the goal for organizations at scale. This kind of information brings insights into key business questions, such as:

- "How much is a customer costing us?"
- "Am I charging the right amount for my SaaS product?"
- "Can I standardize my costs across products or teams?"

Doing this, in practice, can be notoriously difficult. You'll likely want to implement this strategy as you scale, so it'll pay to think about how to do this in advance. If you're using tag management, for example, you'll need to make sure your tag hygiene is excellent and built to support this kind of allocation.

Cost Visibility (Cont...)

Strategy Two:
Organizational Mapping (Cont...)

Risks

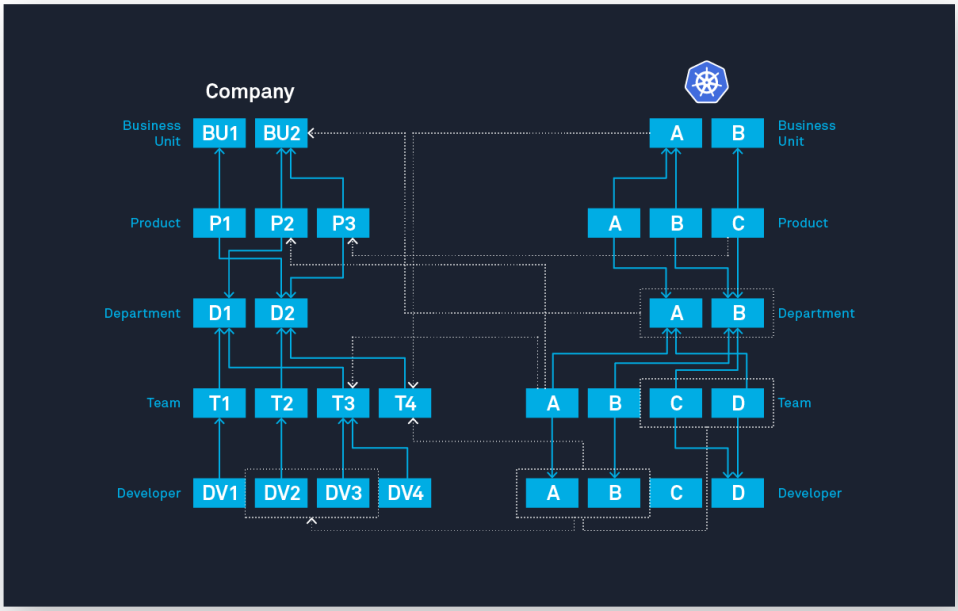
The biggest risk with this strategy is in implementation. Successful organizational mapping requires business context at all levels of the organization and a strong ability to understand and get transparency into your Kubernetes costs at any granularity. In addition, you need to be able to map each Kubernetes cluster, node, namespace, and workload to the business context you're looking at. This can require a large investment in maintaining this mapping, even across cases such as a company reorganization or an infrastructure change event.

When It's Most Useful

Organizational mapping is most useful for organizations that:

- Need accurate chargeback and showback across a vary of different use cases or business contexts.
- Want to use cost allocation data as a core metric to inform and speed up business decisions.
- Have governance requirements that they need to methodically execute on to ensure compliance.

We at Harness have found that mid-market companies and enterprises can effectively leverage this strategy, and that they both want a view of cloud costs that more closely correlate to business goals.



Cost Visibility (Cont...)

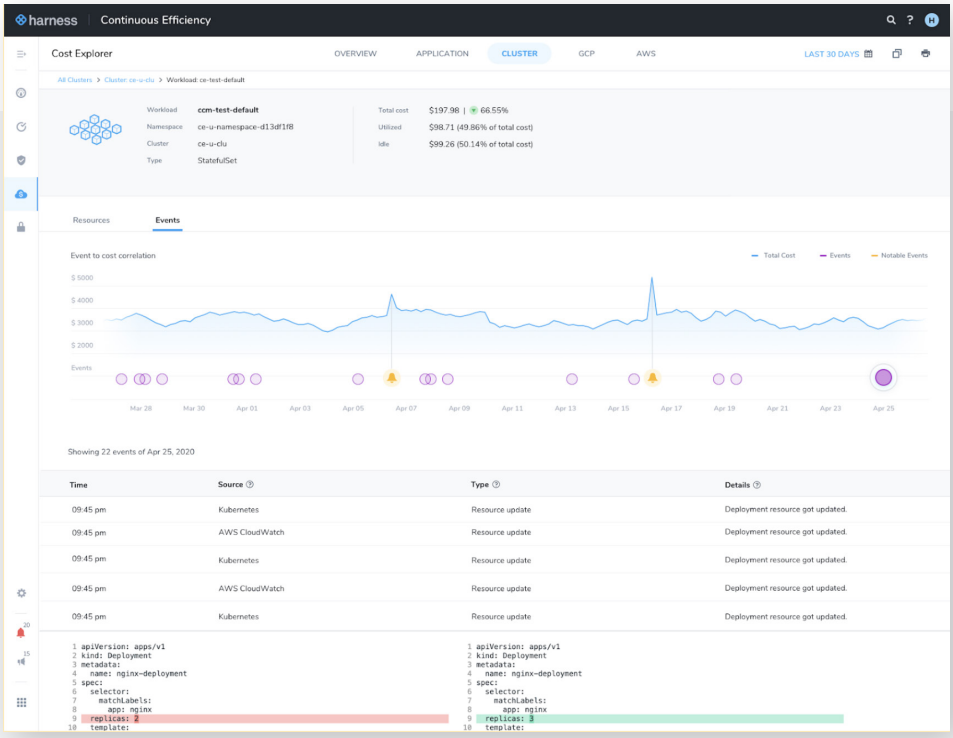
Strategy Three:
Deployment Correlation

Deployment correlation is the practice of mapping costs to individual engineering deployments to Prod, Dev, Test, Staging, or QA. With insight into which cloud resources are associated with deployments, organizations can start to answer questions, such as:

- "Can I get the root cost analysis of an unexpected change in costs?"
- "Can I attribute profit/loss to an individual feature change?"
- "Are there efficiencies I can create across our deployment strategy?"
- "What is the correlation between deployment frequency and our cloud costs?"

We at Harness find that this strategy is emerging as a new methodology for cloud cost visibility, including visibility into Kubernetes. Organizations are now looking to understand the impact of feature changes to their costs and even more granularly understand the different variables involved in cloud spend. For example, organizations want to understand when there is an unexpected change in their cloud costs, when it began, and which code deployment introduced the issue. This allows them to quickly triage potential cost snowballs and remedy the specific deployment.

Of course, attributing Kubernetes costs to deployments requires an intimate knowledge of the continuous delivery pipeline and how Kubernetes resources are being allocated within each deployment. If managing CI/CD pipelines wasn't already enough, now you have to combine that with your cost visibility strategy! On the tail end of the process, however, you have a complete view across all dimensions of your costs, which can be very powerful indeed.



Cost Visibility (Cont...)

Strategy Three: Deployment Correlation (Cont...)

Harness recognizes the value of such an approach. As a software delivery platform, Harness can tie into your CI/CD pipelines and pull this information to give you a view of how deployments are affecting the costs incurred by your Kubernetes workloads, including showing the exact change that was made to impact the change in costs.

Risks

The primary risk with this strategy is the time and effort required to implement. With the level of involvement going beyond just your cost management strategy and spilling into your continuous delivery pipelines, this is no small feat. While the rewards can be great, the cost of building and maintaining such a solution can be a deterrent. Since many tools struggle to solve it, there are some that can help correlate your CI/CD to your cloud costs.

When It's Most Useful

The organizations that find deployment correlation more useful are those that:

- Want to associate engineering and infrastructure changes with changes in cost, including anomalous cost spikes.
- Use cost as a baseline metric to drive marginal improvements to the business.
- Have a goal of increasing cost efficiency among engineering teams.

At Harness, we've found that there isn't necessarily a select group of organizations that are able to best leverage the deployment correlation strategy. Rather, the driving factors are either a desire to further improve cost management, or to further refine the software delivery process. While this typically is something organizations deal with as they scale, even smaller organizations find value in creating a scalable process here early on.

Summary

These three strategies each provide a layer of information leading to the ultimate visibility into your costs. Whether you use one or all of them, the first step to any cost optimization exercise is to see where the money is going. It compares well with personal budgeting: how well can you decide where to spend (or not spend) money if you don't know where you're spending it? There are a variety of methodologies for gaining cost visibility, and no matter what you choose, the biggest thing is making sure you understand what it is that you need to see before implementing anything.

Cost Savings

Once you have visibility into your Kubernetes costs, you can start to effectively save on them. It's hard to save money personally if you don't know where it's going. With visibility into spending, you can decide where to make cuts.

One of the biggest difficulties in implementing cost savings is the variety of teams that work on delivering and maintaining applications. Infrastructure, CloudOps, and Platform teams manage clusters, while Application and DevOps teams manage the services and applications deployed on those clusters. The work spans across these teams for optimal resource efficiency, but also introduces more complexity with so many teams and infrastructure consumers involved.

Let's take a look at some of the ways in which you can save money on Kubernetes.

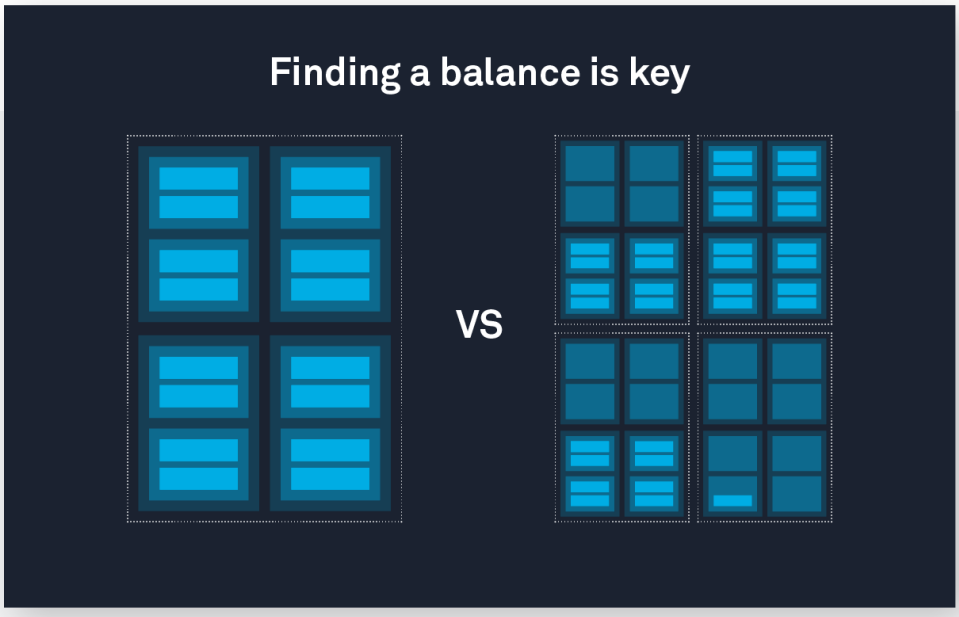
Examples of What You'll See

With a good view into cost savings, you'll be able to find a variety of savings opportunities. Some of these include:

- Finding unused or "zombie" clusters and nodes that you can kill.
- Rightsizing based on actual node resource requirements versus initial assumptions.
- Increasing pod density to optimally use node resources and reduce idle costs.
- Storage being underutilized or unattached.
- Allocating appropriate resources to workloads with high needs and requests.

Strategy One:
Cluster Downsizing

Fewer clusters mean less compute cost. However, removing entire clusters isn't the only way to reduce cluster costs. You should have visibility into how your Kubernetes resources are being utilized, and you should be able to understand which resources are unallocated. These are the low-hanging fruits that let you save on Kubernetes right away.



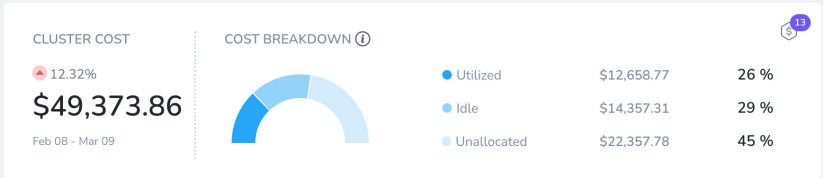
Cost Savings (Cont...)

Strategy One:
Cluster Downsizing (Cont...)

In a lot of ways, downsizing is a fancy way of saying "find what you provisioned and paid for that nobody is using, and get rid of it." Whether it's entire clusters, or nodes within clusters that are overprovisioned, trimming the fat on unallocated Kubernetes resources is a quick way to reduce costs.

The simplest way to get started here is by hooking up Kubernetes to Prometheus (or similar tools) so that you can get your monitoring set up. From there, you'll be able to see your utilized, idle, and unallocated resources. For this strategy, you want to look at your unallocated resources and cut down on the ones you really don't need.

Using good Kubernetes cost management tools can also provide you this information at a glance so you can minimize the toil involved with a Prometheus setup. For example, Harness provides this visibility out of the box so that you can see your breakdown of utilized, idle, and unallocated costs by cluster. You can also see, at a workload level, how much memory, CPU, and storage are costing by the same metrics.



Risks

Cluster downsizing, especially if you're killing entire clusters, can be fraught with a lot of risk and requires clever engineering to make it work without interruptions to the service. Imagine if you had five people to do a job and then you cut it down to two. How would that affect your ability to get things done? You have to be clever about how you work around those constraints while making sure you don't drop the ball on the things that need to be done.

As with all infrastructure changes, you'll also need to ensure that the unallocated capacity isn't a result of a need that someone else has before you kill it. Especially if you are doing this manually and without an autoscaler, it can be troublesome to spin up or down resources for new nodes or new clusters.

When It's Most Useful

Downsizing is appropriate for all. The beauty of Kubernetes is that you can define the need to the control plane or master node and the result will magically be taken care of. However, there is a good chance that waste will accumulate, like with all things, and you'll have resources that become "zombies" - provisioned at one time, but then left unused or forgotten.

Given the breakdown between cluster and workload managers, cluster-level optimizations like this are generally most appropriate for Infrastructure, CloudOps, and Platform teams, though of course any team that governs or manages the provisioning, usage, and operation of the clusters can use this.

Cluster downsizing is a great strategy for organizations at all levels and is one of the easiest strategies to implement and see strong returns. It's recommended that you use an autoscaler to get the most out of this strategy.

Cost Savings (Cont...)

Strategy Two:
Workload Rightsizing

If downsizing is getting rid of unallocated resources, then rightsizing minimizes idle resource costs. Instead of looking at which resources are completely unused, in this scenario you're looking at which resources are being underutilized, usually at the pod level. This allows you to move workloads around and create a better profile for the compute resources you need to provision for the node.

Rightsizing often results in increased pod density, which better optimizes the use of resources across the node. To achieve this, you first need to understand historical usage or workload patterns. With this knowledge, you can understand that if average CPU utilization is 40%, then maybe you don't need the level of compute initially thought, and you can change the configuration to use a compute resource with a smaller cost footprint.

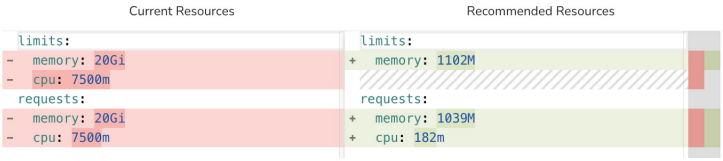
The other side of the coin is making sure is making sure that nodes have the appropriate resources allocated to them. In the above example, what if it turns out your CPU utilization is 40%, but you consistently run out of memory? In this case, you can't just get a smaller resource. You have to change your request and limit profile entirely to decrease relative CPU utilization but increase memory parameters. However, it's more common that both CPU and memory are overprovisioned. In either case, you want to ensure you select the right worker nodes for the workloads that need to be handled.

Cost Savings (Cont...)

Strategy Two:
Workload Rightsizing (Cont...)

As with downsizing, you want to first get visibility into utilized, idle, and unallocated costs, which is done by hooking into Prometheus (or similar tools) and visualizing the usage. For this strategy, you want to look at your idle costs and decide what the best path forward is in terms of resizing resources or moving workloads around. These basic steps will set you on the path towards minimizing idle resources and being more cost-efficient.

A tool like Harness can dramatically reduce the effort required to find rightsizing opportunities and figure out what the right requests and limits should be. By doing this, your historical data is leveraged to automatically generate recommended resource profiles, so all you need to do is go and make the change.



Risks

The biggest risk to implementing a good rightsizing strategy is poor understanding of the data. Without good visibility into how your resources and workloads are performing relative to the requests and limits you've set, it becomes impossible to rightsize. A key consideration that can be easy to forget is for application-level metrics in addition to rote workload-level metrics. For example, you should make sure to consider application-level metrics like JVM heap sizes for JVM-based microservices. Are you appropriately sizing for these needs, too?

You'll also want to make sure, as with downsizing, that you're not stepping on any other toes in the case of shared resources across the Kubernetes environment.

When It's Most Useful

Rightsizing can end up being very technically-involved. As such, it's most useful at organizations that have a good way to track their utilization metrics, which can be as simple as plugging into Prometheus to capture core infrastructure metrics. This strategy is good for teams that have the know-how to move workloads around without breaking things, and can effectively separate expected usage versus the overhead safety net to ensure both performance and cost considerations can be met.

Given the breakdown between cluster and workload managers, workload-level optimizations like this are generally most appropriate for Application and DevOps teams, though of course any team that governs or manages the provisioning, usage, and operation of Kubernetes workloads can use this.

For teams that are looking to more optimally provision and use existing resources, rightsizing is a solid cost savings strategy. It's recommended that you use an autoscaler (specifically Vertical Pod Autoscaler, in this case) to get the most out of this strategy, since autoscaling takes care of the macro-level optimizations, leaving you to focus on the more micro adjustments that come with rightsizing without too much additional overhead to cut through.

Cost Savings (Cont...)

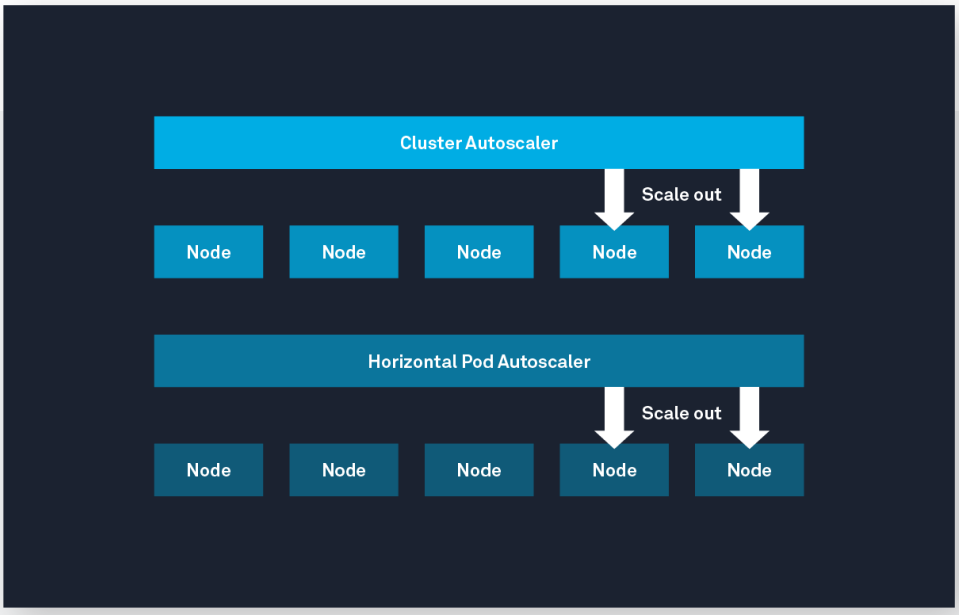
Strategy Three:
Autoscaling

If you don't have to keep track of and optimize your Kubernetes footprint manually, why not go for an automated approach? Autoscalers provide you the ability to specify the conditions under which more resources should be provisioned, or when resources should be terminated. In addition, you can set the floor and ceiling for resource provisioning so you don't inadvertently do too much in either direction.

Kubernetes provides autoscalers that let you autoscale your workloads or pods (Horizontal Pod Autoscaler, Vertical Pod Autoscaler, Kubernetes Event-Driven Autoscaler), as well as autoscaling your clusters or nodes (Cluster Autoscaler).

You'll want to use workload or pod autoscaling through HPA, VPA, or KEDA when you want to autoscale workloads based on defined metrics. If your usage for a pod or workload crosses a threshold compared to the target metric, things can be scaled up, such as using more pods or increasing resource limits. Similarly, if usage is very low compared to the target metrics, things can be scaled down. The ability to scale pods and workloads is limited only by the resources made available to the node in which these reside, meaning if a node's resources are at their limit, your autoscaling stops there.

Autoscaling a cluster or node makes pod scaling more effective. While pod scaling affects the scaling and provisioning of resources within a cluster or a node, it effectively determines the overall amount of resources available to all pods and workloads. Where pod scaling makes the most of available resources, node scaling determines the amount and type of available resources. Cluster Autoscaler detects when pods are in a pending state (waiting for resources) and scales up the number of nodes to add pending pods to. It also detects the opposite, when nodes are no longer needed, and scales down resource consumption.



Cost Savings (Cont...)

Strategy Three:
Autoscaling (Cont...)

Risks

Bad autoscaling policies are the bane of any Kubernetes optimization effort. If limits are set incorrectly or conditions are met through strange edge cases, it can result in out-of-control autoscaling that causes a dramatic increase in costs. At the same time, autoscaling is a great litmus test or even leading indicator of something going wrong in your assumptions. For example, you might see rampant non-anomalous scaling that signifies user growth, or that costs spiral out of control, and that automation might be the first place to look.

When It's Most Useful

Autoscaling is useful for any organization that needs some form of cloud automation. If you're applying autoscaling policies in AWS, GCP, or Azure, chances are you'll want to leverage autoscaling in Kubernetes. With the right limits set in place, and controls around cleaning up any issues that arise (such as cost snowballing or zombie resources), autoscaling is a tremendous step forward for any organization looking to more optimally spend money using Kubernetes.

Summary

We've seen Harness customers that have reduced their bill by as much as 80% using these Kubernetes cost saving strategies. While these strategies can be individually used to reduce costs in Kubernetes, they're most effective when you can leverage all of them. The key thing to remember in reducing costs is that it's notoriously difficult without good visibility. As you can see in each of these strategies, having a good understanding of what's going on is critical to maximizing the returns and mitigating risk associated with any infrastructure change.

Cost Forecasting

Once you have visibility into your costs, you can harness that data to predict what your costs will be in the future. But there's no avoiding it: accurately forecasting is all about statistical analysis. Since it's inherently a projection of the future, the best that you're able to do is use your knowledge of existing data to project what it will look like.

Cost forecasting typically takes two forms:

- Forecasting what you'll spend at some point in the future.
- Predicting whether current cost patterns will exceed budgeted spend.

With historical data, you can predict expenditure at a given point in time, allowing you to do the following:

- Avoid overspending and cost surprises.
- Simplify budget and capacity planning.
- Implement better governance (e.g. budgets are soft limits, quotas are hard limits).

Cost Forecasting

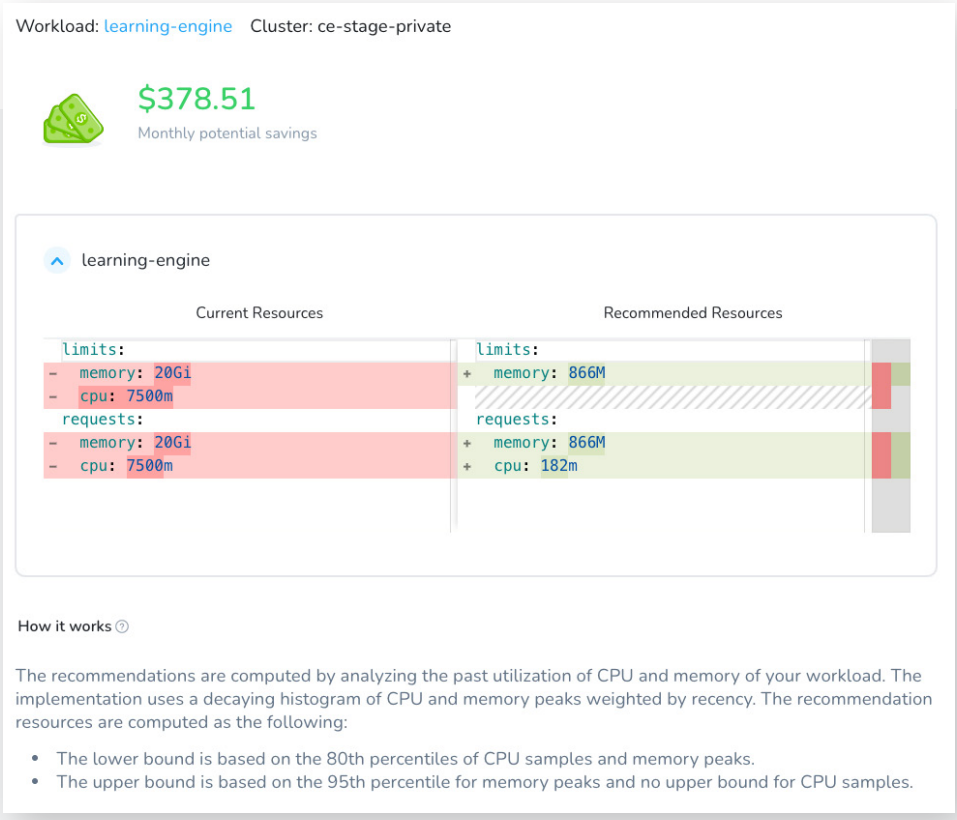
Examples of What You'll See

With cost forecasting capabilities, planning for the future and creating a strong feedback cycle between the three steps of your Kubernetes cost management strategy is greatly improved. When you can forecast, you can do the following:

- Predict Kubernetes costs to accurately budget for future development.
- Proactively avoid budget overspend by predicting cost fluctuations, such as seasonality, and provisioning resources accordingly.
- Project and enforce budgets against Kubernetes consumption of cloud resources.
- Map costs to teams that own them so they know what they'll be responsible for, thereby improving transparency, visibility, and accountability.
- Avoid chargeback surprises by creating channels of communication that align teams around cost changes that may result in budget overruns.

Strategy One:
Soft and Hard Limits

This is the primary strategy used by organizations that do not have a good way to predict their costs, though it is certainly used by all organizations. Setting limits on spend is an easy way to limit usage of resources and ensure you don't accidentally go bankrupt. However, it can be a very reactive and inaccurate method that is disconnected from the reality of the business.



Cost Forecasting (Cont...)

Strategy One:
Soft and Hard Limits (Cont...)

Let's say you are using a simple Kubernetes autoscaler to provision more resources as demand increases or, in user terms, provision more resources to run the app as users come onboard. However, more users onboard than you had planned for. Hard limits, such as resource quotas, were defined in the control plane to keep costs within a budget, and now you're blocked from spinning up more resources. What happens? Either your app crashes, or some users aren't able to access the app. That's never a good outcome.

In an ideal situation, you want to be able to set that limit, but have it set intelligently so that you make efficient use of infrastructure and don't run into walls. When intelligent boundaries are set, you will have more leeway for unexpected overhead and still keep your infrastructure and cost issues to a minimum.

One way to set more intelligent boundaries is to use a tool, such as Harness Cloud Cost Management, which analyzes historical usage pattern data and lets you know what requests and limits to set for your workloads.

Risks

As illustrated above, you want to avoid limits that are set too low and may create problems for users or for the balance of the infrastructure. Without good forecasting, you'll tend to err on the side of setting limits too high to optimize for performance, and it ends up in missed efficiency opportunities.

When It's Most Useful

Setting limits via resource quotas is useful for all organizations across the board. Limits are a great way to ensure you don't spend more than you can afford to, and that you're staying within reasonable expectations. However, be careful not to rely on limits alone as a way to manage and forecast your costs.

Strategy Two:
Regression Analysis

Regression analysis is an advanced statistical approach to cost prediction. Now, you're getting into the weeds of what it takes to actually predict cost, the variables involved, and how they play with each other to inform an accurate projection of future costs.

With regression analysis, an algorithm covers the majority of use cases for what impacts costs, and by plugging in values to the equation, you can predict what the cost will look like at a given point in time. You will need to have an intimate understanding of what contributes to cost and what affects those contributors, which is no small feat if you're not an analyst or statistician. A team will be needed team to do the analysis and create the "plug and play" equation, or visualization, of your cost forecast.

Cost Forecasting (Cont...)

Strategy Two:
Regression Analysis

The key here is that regression analysis is powerful, but it often limits you to a point in time – you can make discrete predictions of what will happen and adjust accordingly, but you can't continuously come back and see how the projection is changing. Doing the analysis itself takes a long time, and because you have a manual equation, you need to come back and do the math again every time you want to project your costs.

Risks

In regression analysis, you always take on the risk of being wrong. What if you're correlating the wrong variables in the wrong way and your forecast is way off? If you act on those projections, suddenly you might be wildly over- or under-budgeting for the upcoming time period, neither of which is ideal. The goal of forecasting is to paint an accurate picture of the future so that you can plan for it. Often, this is in the realm of at least 80% accuracy, but the higher the better.

Another risk you run is that doing the analysis is a point-in-time exercise. The typical way it's done is that a project is commissioned, a team comes back with a number, and you plan based on that. But what happens if some major event occurs that completely changes the projection? You can do a best, worst, average case analysis, but it still doesn't capture the full picture of how things change day to day. To offset this, you can create a running spreadsheet that you can refer to daily or at any other measure of time, but unless you can pull in the people who need to see it and the people who need to act on it, it may become a fruitless endeavor.

When It's Most Useful

At the end of the day, regression analysis is most useful when your primary use case is a one-time future projection to enable planning. Most typically, we've seen organizations care about regression analysis when they're in a position to start projecting into the future and plan ahead. This is usually at the mid-market to enterprise stages of an organization, though this doesn't exclude others who may need the same capabilities. Arguably, all organizations need to use this method at some point.

While it can take a lot of upfront investment to perform regression analysis correctly, it's very useful whenever you want to project what costs will look like at a given point in time, allowing for budgets to be adjusted accordingly. It is most effective when paired with visualizations and action mechanisms to work off of the analysis results.

Cost Forecasting (Cont...)

Strategy Three:
Machine Learning

Think of leveraging machine learning as pumping steroids into regression analysis. Machine learning provides you the power to create incredibly accurate regression models that find all of the contributing factors, what affects them, and what effect they have on the final result. And all of this can be done with minimal human intervention, and on a continuous basis.

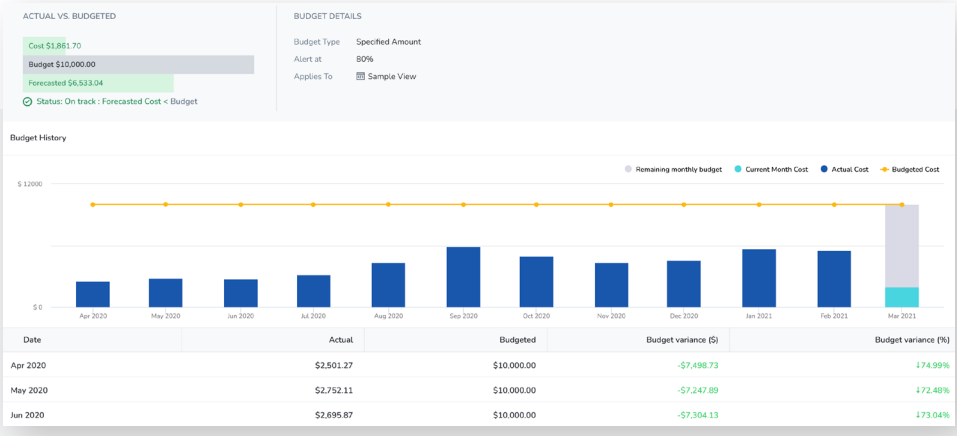
Imagine if you could see how projections change daily so you could adjust allocation of funds before you run into problems. And imagine if you could do this just by looking at a dashboard instead of a bloated spreadsheet, all based on your exact business context. That's what machine learning for cost forecasting lets you do.

However, implementing a machine learning cost forecasting model is no easy feat. Large organizations sometimes devote entire teams to solving the problem, and at the very least, you need to have a strong ML engineer that can architect, build, and validate the system. There are lots of good open-source ML models out there that specialize in projection, but you'll need to adjust those for cost forecasting specifically and train them on your specific business context.

Because it's so powerful and provides so much value, Harness includes machine learning-based forecasting out of the box. Simply connect your billing data and let the engine do the rest. You can group together resources that are relevant to you, set a budget, and see your forecasted costs versus your budget. This enables you to be more proactive about staying within budget, and helps you do better capacity planning and budgeting for future cycles.

Risks

This is where it really starts to become a question of how accurate your forecasts need to be, how often they need to be run, and whether they're a core part of your organizational needs. Is it worth it to you to invest in building an ML-based cost forecasting solution, or can you live with another forecasting strategy? Of course, you can buy solutions that will do this, but you'll want to ensure they are able to work for your business context. You'll also want to consider the granularity at which you need to be able to forecast costs.



Cost Forecasting (Cont...)

Strategy Three:
Machine Learning (Cont...)

Risks (Cont...)

In addition to developing the solution itself, how will you be using it? Will it be purely to forecast costs and do budget planning, or can you stretch and do other things with continuous forecasting capabilities? For example, can you devise ways to proactively alert teams as soon as a projection indicates they're on track to go over budget? Consider the risk here of not fully utilizing your investment in machine learning, especially since it's not uncommon for solutions with poor ROI to be scrapped down the line.

When It's Most Useful

Machine learning-based cost forecasting is at its most useful when paired with effective visualizations and when action mechanisms are tied to the forecasts. Think about it: for something that continuously spits out projections, it's easiest to see those changes visually; and when things change constantly, you need to be able to adjust the outputs tied to the original analysis. More tangibly, you probably want a dashboard tied to your relevant context and desired granularity; and you want a way to make changes, or at least alert decision makers that something has changed and they need to act on it.

While the level of power and detail afforded by an ML solution typically seems like something only the largest or most complex of organizations could utilize, the beauty of ML software is that it can be used by organizations of any scale – it's just a way to utilize vast troves of data in the most insightful way possible. Any organization that wants to be data-driven and create accurate cost projections will find this strategy useful.

Summary

Predictable costs in Kubernetes and the cloud are the holy grail for many involved in the financial management part of the cloud. Whether it's a high-level finance need or a question of budget for engineering teams, being able to know what you need to spend makes everyone's lives easier. The same way that organizations knew in the data center what would be spent on infrastructure, they want to nail down the same for cloud costs, which can be elusive.

Simplifying Kubernetes Cost Management with Harness

Implementing any or all of these cost management strategies for Kubernetes can be time-consuming and risky. Depending on your needs, timelines, and bandwidth, it could be that you leave money on the table that you know you could be saving if only you could get around to it. This is where using good tools comes in handy. There are lots of great tools out there already for cloud cost management, but what about container costs, like with Kubernetes?

Simplifying Kubernetes Cost Management with Harness (Cont...)

That's where Harness comes into the picture. Harness Cloud Cost Management is built with container cost optimization as the primary focus. The same way that complexity in cloud costs is abstracted away by existing tools, Harness Cloud Cost Management paints an easy-to-understand picture of your Kubernetes costs.

In particular, Harness Cloud Cost Management simplifies the entirety of the overarching cost management strategy, providing a solution for cost visibility, cost savings, and cost forecasting. By using Harness, you're implementing all three legs of cost management for Kubernetes and creating a strong feedback loop, letting you rest assured that you're doing right by both performance and cost.

Cost Visibility in Harness

Harness provides you the ability to implement and use each of the cost visibility strategies without additional legwork, and a customizable view into your costs, no matter how you're organized.

- Transform cost data at a macro and micro level, and get hourly granularity into your Kubernetes costs and where they come from.
- Provide idle and unallocated visibility of resources to enable quick understanding and early wins in cost savings.
- Organize costs and map them back to owners via Perspectives.
- Break down and correlate costs by utilized, idle, and unallocated resources, as well as engineering changes like software deployments, config changes, and autoscaling with Root Cost Analysis.
- Proactively identify, alert, and manage even the smallest of cost spikes before they snowball into a surprise on your cloud bill at the end of the month with Anomaly Detection.

Cost Savings in Harness

- Tie into your CI/CD pipeline to take your resource recommendations and more closely match CI/CD provisioning process to optimal patterns based on your usage.
- Get data-backed recommendations for Kubernetes resource request and limit changes and optimize the usage of infrastructure.
- Use Perspectives to view savings for specific contexts including teams, projects, business units, and nodes.
- Run "what if" scenarios to optimize between performance and cost and analyze the impact of each.

Cost Forecasting in Harness

- View cost forecasts for specific contexts including teams, projects, business units, and nodes using Perspectives.
- Create and enforce budgets that are backed by your historical usage patterns.
- Use continuous forecasting updates to proactively respond to cost changes before they result in budget overrun.
- Plan in advance the kinds of resources you'll need to provision based on forecasted cost and known engineering needs.

Simplifying Kubernetes Cost Management with Harness (Cont...)

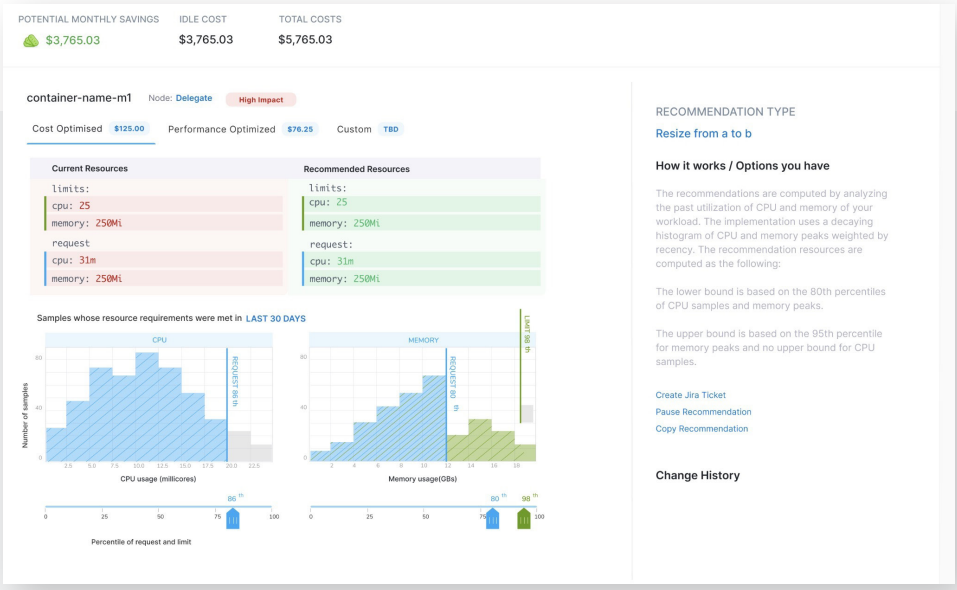
Running Kubernetes "What If" Scenarios: An Industry First

Resource sizing recommendations in Kubernetes shouldn't be limited to optimizing for cost. What if you need to optimize for performance? Or what if you're tasked with finding the right balance between performance and cost depending on your key needs and metrics?

Harness Cloud Cost Management lets you perform what we call a "what if" analysis for your Kubernetes resources. By default, resource optimization recommendations show both a cost-optimized and a performance-optimized resource profile. If that's not enough, you can create a custom lens through which to tune optimization recommendations to balance cost and performance.

Armed with this analysis, you'll be able to leverage Harness to create the best Kubernetes resource profile based on your organization's needs:

- Optimize workloads based on performance, cost, or the right balance between them.
- Run "what if" scenarios to change CPU and memory thresholds to view their impact on cost.
- Illustrate the coverage of CPU and memory samples based on historical patterns of usage.
- Account for seasonality based on changing time periods to analyze how the recommended configurations are impacted.



In Conclusion

At Harness, we have the opportunity to see how hundreds of organizations implement cost management strategies. We've consistently seen organizations of all sizes struggle to get deep visibility into their costs, don't know how or why they should optimize their infrastructure, and repeatedly run into budgeting issues. As organizations deliver more software faster than ever before in the cloud and by using Kubernetes, it becomes imperative to create a solid foundation for understanding and stemming the accompanying costs.

Ultimately, cost management in both the cloud and Kubernetes requires effort that branches across an organization's teams, from finance and engineering to IT and operations. Determining and implementing the correct strategy to meet the needs of all these stakeholders is a worthwhile investment at any organization. In fact, even without tools to aid them, many organizations take advantage of these exact strategies in flavors best suited to them.

While these cost management strategies may not exactly meet your needs, feel free to use them as a basis for formulating your own strategy to meet the growing and complicated needs around Kubernetes cost management.

Author Appendix

Written By:

Bhargav Brahmbhatt, Technical Product Marketing Manager at Harness
Bhargav is a technical product marketer at Harness. With a background as a software engineer, he's invested in bringing the next generation of cool engineering tools (and toys) to the people.

Reviewed By :

Harish Doddala, Senior Director, Product Management at Harness
Harish has led product at high-growth startups and large companies such as AppDynamics, UiPath and VMware. At Harness he works on building and scaling new products, most recently building and launching Harness Cloud Cost Management for engineering teams.

Puneet Saraswat, Director, Engineering at Harness
Puneet is the engineering lead for Harness Cloud Cost Management. With over a decade of experience in networking, distributed systems, and Azure, he brings some serious know-how in how to build the best software for engineers. And, having built Kubernetes Deployment support for Harness, he's a tried and true Kubernetes wizard.

Jim Hirschauer, Senior Director, Platform Marketing at Harness
Jim started his professional journey as an Aerospace Engineer, but his love for computers won. He held roles such as Systems Administrator, Monitoring Architect, and Solutions Engineer before jumping to the Evangelism side and finally, into Product Marketing here at Harness, where he manages a team and gets ship done.