

FinOps and cost management for Kubernetes



Table of contents

03

Introduction

04

Why companies need to track and analyze Kubernetes costs

06

Challenges of analyzing Kubernetes costs

08

Why it's important to analyze Kubernetes costs

09

How to track and manage Kubernetes costs

11

How to bring visibility to Kubernetes cloud costs

13

How to monitor Kubernetes workloads

14

5 main steps to provide visibility to Kubernetes environment

15

How to optimize IT costs in a Kubernetes infrastructure

16

Pod rightsizing

17

Node rightsizing (or virtual machine rightsizing)

18

Autoscaling (horizontal pod autoscaling, vertical pod autoscaling, and cluster autoscaling)

18

Rebalancing fragmented nodes

19

Using cloud discounts options

20

Kubernetes performance issues and ways of remediation

22

Top Kubernetes performance issues

23

How to address Kubernetes performance issues

27

What is the difference between calculating unit economics for Kubernetes and public or private clouds

30

Final thoughts

33

About the author

Introduction

The advantages of Kubernetes infrastructure like portability and scalability, its open-source base and the ability to increase developer's productivity have made container technologies a popular choice for many companies, and Kubernetes has become the standard for running container-based apps across clouds. More than [80% of companies today run containers](#) in production and [78% of them use Kubernetes services](#).

As the containerized infrastructure is obtaining widespread adoption and Kubernetes technologies are gaining momentum it's becoming crucial to understand how to get a clear picture of spending on K8s resources, enforce cost optimization opportunities and enhance Kubernetes performance.

The reality shows that it's not enough just to use Kubernetes to get the best value of public clouds. According to a recent [StackRox report](#), about 70% of companies detected misconfiguration in their Kubernetes environment.

A containerized structure creates significant difficulties with cloud transparency, allocation and performance that cause challenges in resource management and optimization.

The whitepaper covers the main Kubernetes infrastructure management challenges, gives technical tips and best practices in order to provide visibility to K8s environment, optimize costs, overcome misconfiguration issues, improve Kubernetes performance and calculate unit economics for Kubernetes clusters.

It will help you build a solid optimization and management strategy for the Kubernetes environment, make a giant leap forward in improving application performance and reduce its infrastructure cost.

Why companies need to track and analyze Kubernetes costs

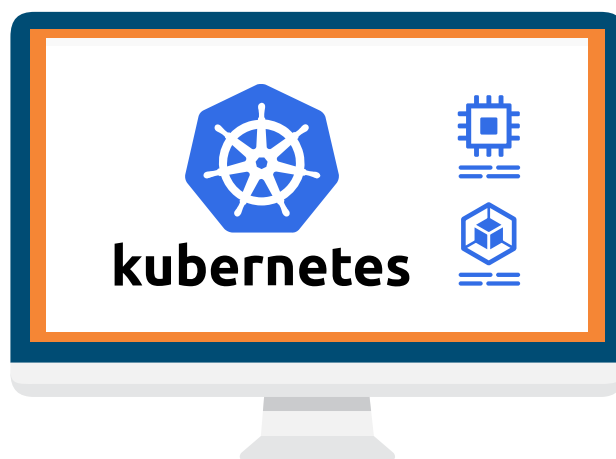


There are two main approaches to virtualization: virtual machines (VMs) and virtual containers, each with its own advantages and disadvantages. In the first case, every machine uses its guest OS, which gives an opportunity to create heterogeneous computing environments on one physical server. Virtual containers, in turn, instead of an OS only have a user environment, which makes it possible to create only homogeneous computing environments.

However, since virtual machines include an operating system, they can be as high as several gigabytes in size — that substantially limits their usefulness in today's agile world. Another disadvantage of virtual machines is that it takes much more time to load the OS and its applications.

Containers are lighter and are generally measured in megabytes; and, what's more important, they are easier to scale and deploy applications. This creates an entirely new infrastructure ecosystem, which means new challenges and complexities. IT companies, both large corporations and start-ups, deploy thousands of container instances every day and thus must somehow manage this overwhelming amount.

Kubernetes, a container orchestration platform for deploying, scaling, and managing containerized applications, was designed as a solution to this problem. Over time, K8s has essentially become the industry standard platform and the flagship project of the Cloud Native Computing Foundation, supported by market leaders: Google, Amazon, Microsoft, IBM, Alibaba Cloud and many others.



Owing to its advantages Kubernetes gains popularity but at the same time it brings significant difficulties to cloud costs tracking and finance management.

Challenges of analyzing Kubernetes costs

Before the widespread adoption of containerization technology, cloud resource allocation and cloud usage optimization was much easier. All that was required in this case was the attribution of specific resources to a specific project or department. It won't be difficult for a FinOps team — if they are part of your IT team — to come up with a cloud cost breakdown and put together a cloud cost optimization strategy. If you would like to learn more about the role of FinOps and what potential benefits it can provide to business, [please refer to an ebook "From FinOps to proven cloud cost management & optimization strategies"](#).



Unfortunately, this approach is absolutely inapplicable for containerization tools in general and Kubernetes in particular. What is the reason why Kubernetes cost is so difficult to define and analyze?

Unfortunately, this approach is absolutely inapplicable for containerization tools in general and Kubernetes in particular. What is the reason why a Kubernetes cost is so difficult to define and analyze?

The difficulty in tracking Kubernetes costs stems from its architecture. At the core of Kubernetes, there is a cluster that consists of numerous virtual (or physical) machines - nodes. On those nodes, containers are deployed and launched - they actually contain various applications.

Now let's say that several departments in your company are working on various applications that run inside containers and, as it happens, share common Kubernetes clusters. It is almost impossible to determine which of the launched applications uses which part of the resources of which clusters, because each of the applications is launched on several containers at the same time.

While calculating the cost of one container is possible and not too difficult in itself, it still takes infrastructure and time, and the complexity grows in proportion to the number of containers used.

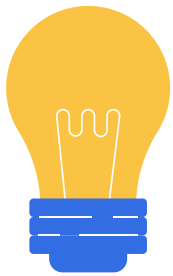
Until now, we considered the situation within the frame of one cloud. But what if your company, like the most of modern IT organizations today, uses the multicloud. In this case, cost monitoring will grow many times over — each of the clouds within this multicloud can have a different service provider, taking on only a part of the total workload because Kubernetes can work with AWS, Microsoft Azure, Google Cloud Platform, Alibaba Cloud and many others.

In addition, the resource intensity of each of your applications can change over time, which imposes additional difficulties in calculating costs. So, the easy-to-use [VPA \(Vertical Pod Autoscaler\)](#) and [HPA \(Horizontal Pod Autoscaler\)](#) tools, which, respectively, automatically adjust the limit on the number of requests to a single container and the number of containers used, become additional hard-to-factor variables when trying to track and manage current Kubernetes costs and, of course, predict future costs.

Another problem is that a container's lifespan is just one day, and functions being run on Kubernetes - down to minutes or even seconds. Again, such dynamism is a definite plus from the point of view of an IT engineer, but it becomes a headache when it comes to cost tracking and management.

Why it's important to analyze Kubernetes costs

All of the above is the flip side of the containerization tools, a price to pay for the extraordinary convenience, flexibility, and agility that Kubernetes brings. Despite the fact that the infrastructure built in this way has a high potential for automatic optimization, the lack of sufficient cost control can nevertheless lead to sad consequences, and, first of all, let the cloud bill spiral out of control.



Why can this happen? For many IT teams, productivity and delivery speed are often prioritized over budget. In this case, the latter may suffer due to unforeseen expenses. This problem cannot be solved by a one-time intervention in the IT department, because technicians will still continue to prioritize high-performance applications and code over expenses.



The autoscaling tools discussed in the previous section are not only a hard-to-factor variable from the point of view of Kubernetes costs control but also a potential time bomb planted under your company's budget if scaling policies are left to chance. If scaling limits are not set correctly or potentially dangerous corner cases are not taken into account, this can result in drastic upscaling that will cause a dramatic increase in costs.

How to track and manage Kubernetes costs

As we already said, it's very difficult to track and manage Kubernetes costs, but there are still numerous techniques that can help you analyze Kubernetes costs — and eventually take them under control.

1. Proper resource labeling

Most probably, you're familiar with cloud resource tagging. When it comes to Kubernetes, the so-called labels are used instead of tags. If you and your engineers take care of the labeling of the resources used, it will greatly facilitate their search and identification in the future. It is important to be smart about this process so that you can later break down your resources by various relevant parameters. Successful implementation of such a strategy will require the active participation of all IT team members; otherwise, this initially good idea can lead to even more confusion.

2. Visualization by means of Prometheus

Open-source monitoring systems like [Prometheus](#) can be great help in visualizing your costs. And competent visualization, in turn, is a giant leap on the way to competent analysis.

3. Proper use of autoscaling

Autoscaling is a killer feature of Kubernetes, and with the help of it, one can easily manage workloads. We already mentioned two of them — [Vertical Pod Autoscaler](#) and [Horizontal Pod Autoscaler](#), but there are actually two more available: [Kubernetes Event-Driven Autoscaler](#), and [Cluster Autoscaler](#), where the former manages the scaling of any container in Kubernetes based on the number of events needing to be processed, while the latter deals with autoscaling on the cluster and node level. That said, it's quite a challenge to make them work together properly — not only should you stick to the numerous best practices, but also fine-tune the settings based on your scenarios.

4. Choosing the right cloud instances

The cost of Kubernetes directly depends on how well the cloud instances are selected. It is important to ensure that Kubernetes pods' resource consumption matches the amount of allocated memory and compute resources of the instance, regardless of which cloud provider is used.

5. Proactive resource management

Underutilized and unused resources are one of the first items to look for direct losses and room for optimization. As mentioned previously, IT specialists prefer performance over resource optimization, so they tend to overuse resources. Despite the tempting prospect to immediately abandon all idle capacities, this must be done wisely, so as not to exclude anything that turns out to be important — the next point follows from this.



6. Hiring a FinOps manager

FinOps can help in solving several problems at once. Firstly, it will pass more responsibility to technical specialists for the financial performance of the company as a whole, and its spending on cloud resources in particular. Secondly, it can become the missing link that can monitor and manage Kubernetes costs on a daily basis so that the scaling of the resources used occurs when it is really needed.

7. Using specialized tools

Last but not least, you can use [third-party Kubernetes cost analysis and management tools](#). It can give you full visibility over your Kubernetes costs, can help optimize IT costs, enhance application performance and engage an engineering team in cloud cost-saving process.

How to bring visibility to Kubernetes cloud costs



Since more and more organizations are expanding the usage of container orchestrators, and Kubernetes is becoming a popular choice for many companies, it's important to understand how to provide full transparency across K8s resources for achieving cost optimization goals and performance improvements.

The advantages of container technologies like portability and scalability and its open-source base have made Kubernetes the standard for running container-based apps across clouds.

Luckily, cloud platforms provide support and help companies of any size to adopt Kubernetes technology. Here is a list of services provided by the major cloud platforms:

- Amazon Elastic Kubernetes Service (Amazon EKS) on AWS
- Google Kubernetes Engine (GKE) on Google Cloud
- Microsoft Azure: Azure Kubernetes Service (AKS)
- IBM Cloud: IBM Cloud Kubernetes Service
- Oracle Cloud Infrastructure: Oracle Container Engine for Kubernetes
- Alibaba Cloud: Container Service for Kubernetes (ACK)



Despite the advantages of Kubernetes, a containerized structure creates challenges with cloud cost transparency, allocation that cause significant difficulties in resource management and optimization.

An effective cloud management needs cost visibility; it's crucial to identify organizational units such as applications, cloud services, asset pools, business units, teams, individual engineers and map them onto cloud costs.

Nevertheless, Kubernetes usually is considered to be a black box. Cost allocation is a complicated task for this containerized technology, even when it is launched by one of the major cloud providers. Kubernetes is often used by hundreds of applications and dozens of engineers as a multi-tenant system simultaneously. Service providers charge and include into a cloud bill the cost for every server instance that makes up a Kubernetes cluster.

Cost allocation in container-based systems makes additional difficulties. Most Kubernetes clusters are shared resources with applications run by many teams, which means that there's no direct cost allocation to a specific container.

How to monitor Kubernetes workloads

Cost transparency across teams, applications and individual pods is a must-have to prevent budget overruns and avoid wastage. Companies often suffer from a lack of systems that identify the cost of each deployment, service or namespace, because Kubernetes doesn't track any data about cost and resource usage.



For Kubernetes monitor and alert purposes companies often use **Prometheus**, built-in Kubernetes software and considered to be one of powerful tools with an open-source base enabling cost transparency of queries and reports.

The Prometheus solution contains the following tools:

- The Prometheus Node Exporter helps in measuring various server resources such as CPU and memory usage of pods, containers and other metrics on nodes in a Kubernetes cluster
- Kube-State-Metrics generates metrics based on the state of Kubernetes objects e.g., node status, node capacity (CPU and memory), a number of desired/available/unavailable/updated replicas per deployment, pod status and so on
- The Prometheus Alertmanager allows to set up alert notifications, thresholds and send emails, trigger a pager or generate a ticket
- Grafana provides companies with the opportunity to visualize Kubernetes resource usage over time and interactive cost exploration

Using [third-party financial management platforms](#) also could facilitate the task of providing cloud cost transparency into Kubernetes infrastructure, improving visibility into shared Kubernetes clusters and their costs. Moreover, cloud financial management platforms often support capabilities that offer multi-cloud management across different platforms with providing billing data on a single dashboard.

5 main steps to provide visibility to Kubernetes environment

Start Kubernetes resource detection

As it is known, Kubernetes does not extract and keep any of the data, which is required for resource usage detection. As a first step towards an efficient Kubernetes cloud cost management it is important to set up a system for tracking shared Kubernetes clusters and storing this data in place.

Attribute Kubernetes workloads

Tagging or resource naming convention is a crucial component of achieving Kubernetes cost monitoring goals. With a deep labeling and tagging it becomes clear what Kubernetes workloads belong to a specific team, organizational unit, individual pods, project or cost center. Consistent labeling and namespace can improve your allocation strategy.

For example, companies can use [Kubernetes annotations](#) in order to automatically attribute costs to the right cost center.

Attach costs to Kubernetes objects

Companies need to implement an accounting system to attach Kubernetes environment costs and related cloud spend to the teams, individual users or business units that consumed these resources. Kubernetes allocation methodology allows your team to break down costs by namespace, label, cluster or service.

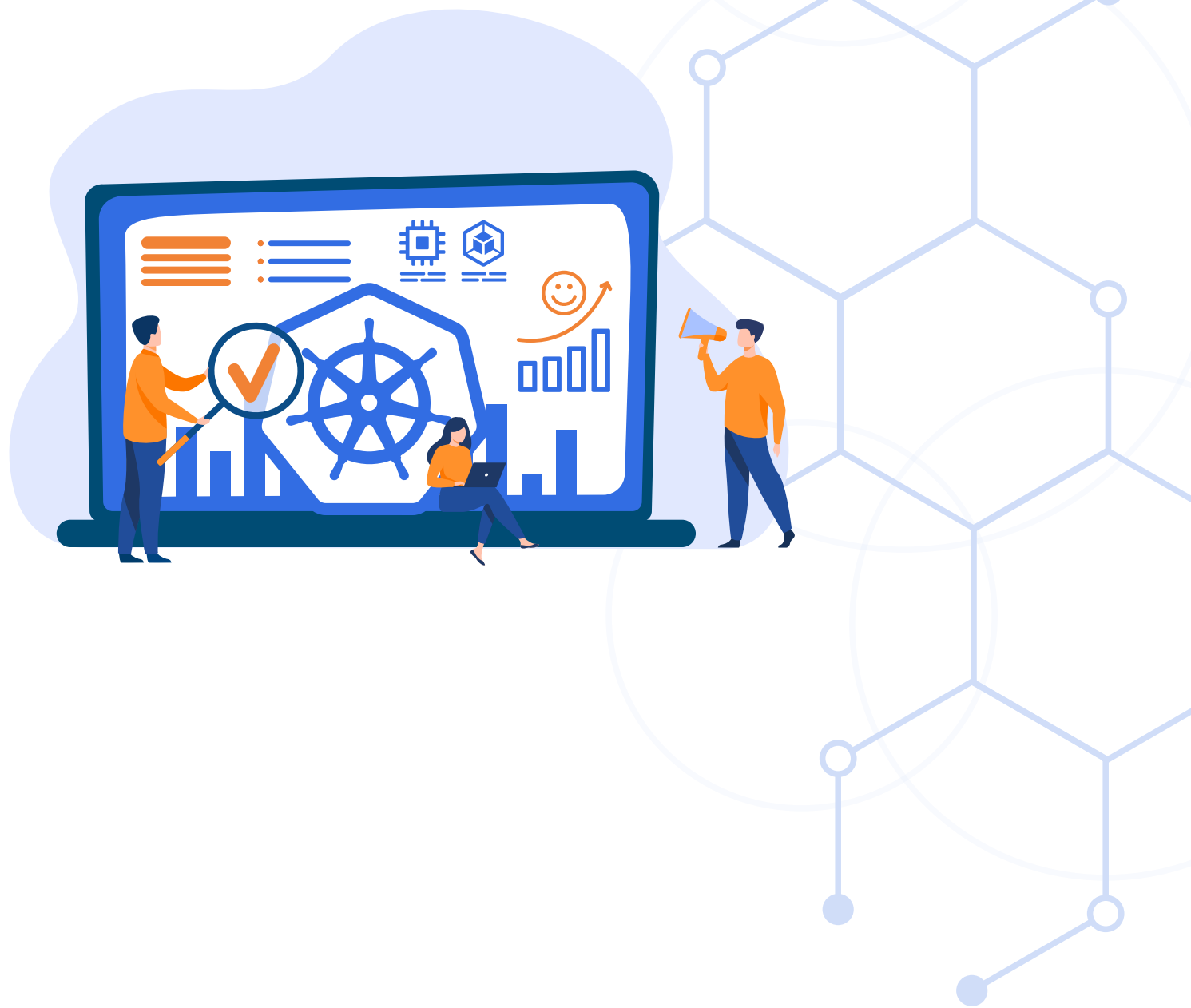
Set TTL rules, notifications and clean up policies for Kubernetes clusters

Smart TTL rules, alerts and budget constraints help to keep all Kubernetes clusters under control and prevent budget overruns. Clean up policies as one of the most important elements of the cost monitoring system enable teams to get rid of obsolete and unused resources.

Create reports

Since the resource usage data gets available, it has become possible to create reports for deep Kubernetes costs analysis to improve the effectiveness of the data usage experience.

How to optimize IT costs in a Kubernetes infrastructure



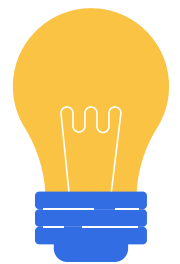
A promising idea to use a Kubernetes container infrastructure for your business is currently gaining momentum. And in case of using K8s, you are most likely to face a question of how to manage expenses in clouds or in an on-premise infrastructure to prevent budget overspending but possess better performance at the same time.

So, there is a list of today's best practices that may be useful in optimizing your Kubernetes cloud costs.

- Pod rightsizing
- Node rightsizing (or virtual machine rightsizing)
- Autoscaling (horizontal pod autoscaling, vertical pod autoscaling, and cluster autoscaling)
- Rebalancing fragmented nodes
- Using cloud discount options (savings plans, reserved instances, spot, etc.)

Let's have a look at all of them to highlight their meaningful features for managing cloud costs in a Kubernetes environment properly.

Cost optimization scenarios for Kubernetes



Pod rightsizing

Each Kubernetes cluster is considered as a set of pods that include containers. Being the smallest deployable elements, pods act as a single block of resources and allow you to run the workloads. You can manage CPU and RAM resources allocated for a specific pod. So, it's essential to pay attention to the right size and schedule of the resources to maintain a cost-efficient and stable process in a pod's environment. For this purpose, while configuring any Kubernetes cluster you need to adjust resource requests and limits to find an optimal amount of CPU and control memory resources per pod. That will provide you with a proper application performance and save compute and memory resources for other pods at the same node.




Otherwise, you may have a situation when you overcommit CPU and RAM but the overall VM or bare metal resources are underutilized and, if you remember, you don't get them for free. This is a transition to the second point of node rightsizing.

Node rightsizing (or virtual machine rightsizing)

Alongside with the rightsizing of the pods in a Kubernetes cluster environment, it makes sense to be sure the nodes you are using for your applications to run are also of the right size and flavor.

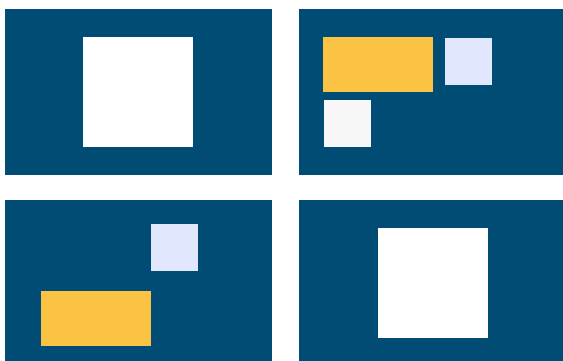
In terms of performance many things depend on a number of factors connected with how many pods per node become active while the workloads are running. Thus, it's necessary to keep it in mind because by default even internal Kubernetes services themselves impose limits on the number of pods per node if it's required.

From the table below you can find pod-per-node limits given by the leading cloud providers.

cloud providers	pod-per-node limits
 Amazon Elastic Kubernetes Service	the maximum number of pods per node: from 4 to 737 (it depends on the node type)
 Google Cloud Google Kubernetes Engine	the limit is 110 pods per node (no matter what type of node is used)
 Microsoft Azure Azure Kubernetes Service	30 pods per node is the default limit but it can be increased up to 250

before node rightsizing

excessive infrastructure and underutilized resources



after node rightsizing

improved resource use and less infrastructure



Autoscaling (horizontal pod autoscaling, vertical pod autoscaling, and cluster autoscaling)

The use of those two aforementioned procedures in a Kubernetes container environment is very important to avoid poor application performance and overspending.

But also in order to reduce the cost of your Kubernetes cluster and provide a swift services adoption to any environment changes, Kubernetes has such autoscaling tools as The Horizontal Pod Autoscaler (HPA), The Vertical Pod AutoScaler (VPA) and The Cluster Autoscaler for managing the number of active pods and nodes. They are best suited for setting the cost-optimized configuration especially in cases of quick reaction to usage spikes and avoidance of workloads instability. Cost saving happens when the tools are properly configured and nodes are shut down or launched at the right time.

Rebalancing fragmented nodes

Over time any active Kubernetes cluster experiences a cycle of inevitable deployments and periodic scale-outs. That leads to repeated pod/node additions and removals and it can cause an inefficient performance and resource fragmentation in a cluster.

Moreover, there is some kind of inconsistency in the Kubernetes scheduler work - they cannot fully predict a future number of nodes and what sizes pods will have. It means that even if some new pods are scheduled for the upcoming process and all the necessary resources requested by them are available, the resources collectively are unavailable at any single node that makes the pod unschedulable. So the extra scale-up is still needed even though the cluster as a whole unit has much more capacity available. The takeaway to get rid of a so-called “pseudo” resource crunch is consolidating all these available fragments of resources together.

That purpose of rebalancing fragmented nodes can be achieved by identifying and migrating a definite set of pods across the nodes in the cluster to combine all the necessary resources together. Especially this procedure of rebalancing unoptimized Kubernetes clusters needs to be performed in large-scale clusters to avoid wastage of resources and reduce cloud costs.

Rebalancing fragmented nodes is an ongoing process and it usually cooperates with pods/nodes rightsizing and autoscaling on the integrated basis.

Using cloud discounts options

As for a Kubernetes infrastructure it's relevant to apply a set of resource purchasing options for the nodes where the cluster runs. Savings plans, on-demand instances, reserved instances, spot instances are available from the public cloud providers.

In general, you can save up to 50% of your IT cost using those programs, so you need to consider them as one of the first steps that doesn't interfere in your Kubernetes cluster or application architecture.

Kubernetes performance issues and ways of remediation



Kubernetes, the open-source container orchestration software, is dominating the world of containerized applications by holding by far the largest amount of its market share. And there are a lot of reasons for that. Kubernetes drastically extends the capabilities of software for containerization-enabled environments such as Docker. It simplifies the management of deployment, network routing, resource utilization, load balancing, the resiliency of running applications and many more.

However, this solution will not work effectively on its own without proper preparation and additional configuration, as every newly created cluster doesn't have an optimal performance by default. There are always subtle difficulties and nuances of Kubernetes' implementation and operation, as well as the problem of suboptimal use of its advantages, which ultimately leads to the loss of money. In this case, representatives of IT teams must possess enough experience, methods and instruments to define misconfiguration and bottlenecks, but at the same time, there is a global shortage of expertise in Kubernetes on the market, because at this time the popularity of K8s is overtaking the level of knowledge about it among technical specialists.

Top Kubernetes performance issues

Based on the [research](#) conducted by Circonus, the top four Kubernetes performance issues are:

- **resource contention for clusters/nodes/pods,**
- **deployment problems,**
- **auto-scaling challenges,**
- **crash loops and job failures.**

It came as no surprise as those issues largely stem from the peculiarities of the technology and the lack of expertise and experience when working with this platform.



At the heart of Kubernetes there is a scheduler that places containers on nodes. Simply put, it's like packing boxes of different sizes with items of different sizes and shapes. From that point of view, the scheduler needs to know the exact capacity of nodes as well as the size of each container being placed on those nodes. The failure to do so results in over-provisioning the nodes and serious performance problems.

How to address Kubernetes performance issues

Based on the research conducted by Circonus, the top four Kubernetes performance issues are:



The most efficient — and, at the same time, the most challenging — way to tackle K8s performance issues is definitely to increase the observability of the platform in order to help you understand which of the collected metrics you need to keep an eye on in order to identify the root cause of certain issues. In fact, Kubernetes provides you with numerous metrics, and the majority of them are an important source of insights into how to use the platform regardless of how you actually run it.

As we already mentioned above, open-source monitoring systems (like [Prometheus](#)) can be a great help in visualizing your Kubernetes costs. And with the help of an exporter standalone program it's possible to translate node metrics into the appropriate format and send them over to the Prometheus server. By installing it onto every node of your cluster, you'll be then able to get access to dozens of metric categories, the most important of which are related to CPU, disk, memory and network usage.

Despite the fact that we have narrowed the range of the studied metrics to four categories, at this stage it will still be difficult for us to understand which indicators are paramount for us. Since Kubernetes is an example of a complex system, we should take the path of simplifying abstractions around the categories of interest to us. Subsequently, this will help us analyze not only node metrics, but in general all Kubernetes metrics.

The most common methods for simplifying abstractions are:

→ The USE Method, introduced in 2012 by Brendan Gregg; targeted at resources in your system:

Utilization — the average time that the resource was busy servicing work.

Saturation — the degree to which the resource has extra work which it can't service, often queued.

Errors — the count of error events.

→ The RED Method (2015), which defines the three key metrics you should measure for every microservice in your architecture:

(Request) Rate - the number of requests served.

(Request) Errors - the number of failed requests.

(Request) Duration - distributions of the amount of time each request takes.

→ The Four Golden Signals (described in the Site Reliability Engineering book by Google) is to some extent a fusion of the above methods:

Latency — the time it takes to service a request.

Traffic — a measure of how much demand is placed on your system, measured in a high-level system-specific metric.

Errors — the rate of requests that fail, either explicitly, implicitly, or by the policy.

Saturation — how "full" your service is.

It turns out that it is not enough to have extensive information about the resources on the nodes in the Kubernetes cluster, it is also important to be able to analyze it. For example, analyzing resources (such as CPU, disk, memory, and network) through the lens of usage, saturation and errors (USE method) can give us an understanding of how resources are being spent and allow us to further optimize and scale their use.

Once your IT team figures out what resources are underutilized and overutilized, they will be able to define the optimal storage limits, the optimal CPU and memory size for cluster nodes, and the optimal nod pools for every node, which in turn will allow them to analyze Kubernetes costs and analyze its performance.



Sticking to best practices

Regardless of how successfully you monitor and analyze your Kubernetes resource usage, there are a number of best practices to follow to help you get the most out of the platform.

→ Optimize your environment for Kubernetes.

Keep in mind that containerized tools were originally designed for a loosely coupled architecture consisting of Stateless applications that process data but do not store anything. Therefore, it is a mistake not to do anything before deploying data-storing Stateful applications and not to adapt the architecture of monolithic applications letting them run on Kubernetes.

→ Use Kubernetes only when it's necessary.

When moving to Kubernetes, remember that it makes sense to run databases and some applications in a virtual machine, and a potential move for the sake of a move can seriously affect performance.

→ Have specialists who know how to work with Kubernetes.

Working with Kubernetes requires system administrators with hands-on experience with the platform, as successfully maintaining this ecosystem of components requires a high level of expertise.

→ Adapt IT processes for Kubernetes implementation.

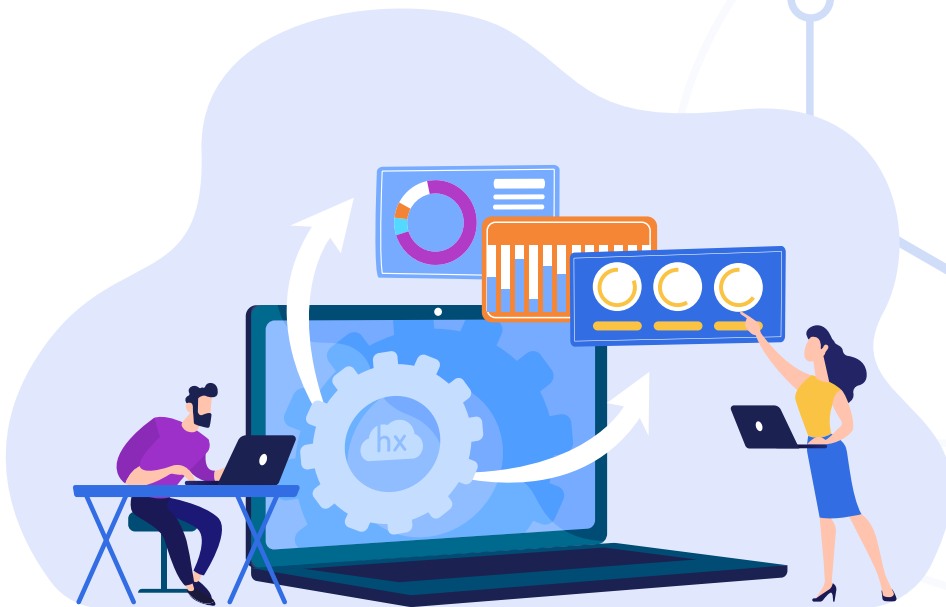
Kubernetes is fundamentally changing the distribution of roles and responsibilities within an IT team. Now, the proper implementation requires a shift to DevOps processes, and developers should accept this methodology and its tools.

Through DevOps, system administrators maintain the infrastructure, while developers support the application from planning and coding to launch, implementation, monitoring, and production. Developers now cannot help but know the infrastructure; they should also understand how their code works in the context of all these new processes.

→ Leverage additional tools that extend Kubernetes functionality without relying solely on out-of-the-box functionality.

In the previous paragraph, we dwelt in detail on the Prometheus-based metrics monitoring system, but this is far from all the functionality that additional services can provide. Also, thanks to various tools, you can optimize the processes of storing application data (Ceph, GlusterFS), collecting and storing logs (Fluentd, Elasticsearch, Loki), autoscaling (Metrics Server, Prometheus Adapter), security settings (Dex, Keycloak, Open Policy Agent) and much more.

What is the difference between calculating unit economics for Kubernetes and public or private clouds



Every company at some point in time starts to calculate and analyze unit economics. It can be done to prepare for a new investment round or to monitor and improve the existing metrics. But if it's not a big deal for private clouds, it may be tricky for public cloud environments like AWS, Microsoft Azure and GCP, or Kubernetes clusters.

Please keep in mind that the end formula for each company is different and depends on your business, product and what parameters you prefer to use. For example, you may use only the cost of the production environment but some companies can also add the R&D costs (including infrastructure) to the equation.



Unit economics for private clouds

For private clouds like VMware, Nutanix or bare metal, calculating unit economics is a pretty straightforward task, because a private cloud environment is not as variable or elastic as public clouds, we know how big our TCO is and we can calculate the economics. Here in the majority of the cases we take the TCO and divide it by the number of customers adding extra parameters into the equation, if necessary, like the cost of ads, marketing activities, datacenter lease etc.

Unit economics for public clouds

For private clouds like VMware, Nutanix or bare metal, calculating unit economics is a pretty straightforward task, because a private cloud environment is not as variable or elastic as public clouds, we know how big our TCO is and we can calculate the economics. Here in the majority of the cases we take the TCO and divide it by the number of customers adding extra parameters into the equation, if necessary, like the cost of ads, marketing activities, datacenter lease etc.



Unit economics for Kubernetes

For [Kubernetes environments calculating](#) unit economics is extremely difficult because you can't map your cloud cost on pods, namespaces, services and applications in your Kubernetes cluster. You know how much you pay for nodes, you can calculate the cost of the whole cluster using the recommendations above depending on if your cluster runs in a private or a public cloud.

But then you need to figure out how to distribute the cluster expenses between pods, services and applications. Also, it is a very complex task to track a pod creation and destruction to properly get historical data and calculate the cost for it. You need to use libraries like Prometheus and Cost Analyzer to get historical data and monitor information, calculate idle resources and divide the cost of Kubernetes nodes across pods at every moment of time. You need to have labels for all their applications, namespaces and services in your Kubernetes cluster so you can properly group the resources and expenses.

Final thoughts



Kubernetes gains widespread adoption for several reasons, among which the following are of particular note: it's scalable, cost-efficient and cloud-agnostic. However, there are certain drawbacks to using containerized tools; one of them is the complexity of cloud costs management.

In addition to the most common challenges such as a lack of transparency and solid optimization scenarios, significant difficulties appear with tracking [shared cluster resource usage](#) and providing a simultaneous shared access to set up an effective collaboration.

The implementation of some key principles will help your engineering team address these issues and overcome Kubernetes cost management challenges.

Bring visibility

Tracking key application and resource metrics, identifying wastage, defining owners, getting a clear picture of the health, price and performance of resources is a fundamental and crucial element for optimization and a cost-saving process.

Optimize IT costs

A properly built and constant optimization process makes it possible to balance cost, reliability and performance. Your team should understand how autoscaling mechanisms work and how to implement other useful cost-optimized configurations for cloud expenses reduction.

Enhance performance

Mismanaged cloud environment is a common problem IT leaders face on a daily basis. An engineering team should correctly define a set of features and configurations such as VM rightsizing, machine types, region selection, CI/CD job resource reflavoring or pod affinity groups for achieving better performance. The optimal choice of storage limits, CPU and memory size for Pods helps to avoid some challenges at one of the first stages of improving Kubernetes performance.

Taking into account Kubernetes costs while calculating units

Getting a true understanding of a company's Kubernetes costs, as a part of calculating unit economics, enables more productivity and ensures profitable business growth. IT teams should overcome the challenges of tracking Kubernetes and bring visibility to cloud cost metrics.

Build a FinOps and cost-saving culture economics

The FinOps methodology aims to build an effective cloud environment. The implementation of the main FinOps principles such as collaboration, long-term optimization and cost-saving processes over the Kubernetes infrastructure has become an integral part of an effective cloud cost management strategy.

Over time, K8s has essentially become the industry standard platform and the flagship project of the Cloud Native Computing Foundation, supported by market leaders: Google, Amazon, Microsoft, IBM and many others.

In order to achieve exceptional Kubernetes performance with application stability and minimal costs, IT teams should bring observability, implement optimization scenarios, correctly set Kubernetes configurations to benefit from containerized technologies and make the most out of Kubernetes environment despite some challenges of this containerized technology.

About the author



Nick Smirnov

CEO, Hystax

Nick Smirnov is a FinOps and digital transformation enthusiast with more than 10 years of expertise working with public clouds and in enterprise software development. As the CEO and Co-Founder of Hystax, Nick has been working with world-renowned managed service providers, cloud services providers and telecommunication companies. Nick is passionate about cloud adoption and is helping companies navigate cloud cost management more effectively with Hystax OptScale. He is helping them benefit from a solution that engages engineers, leverages cloud management and resource sharing to help them gain complete cloud transparency and recognize significant cost savings.

To complement his cloud 'addiction', Nick devotes his additional time to learning several foreign languages, working out at the gym and traveling with his family.