



Complete Guide to Kubernetes Cost Optimization

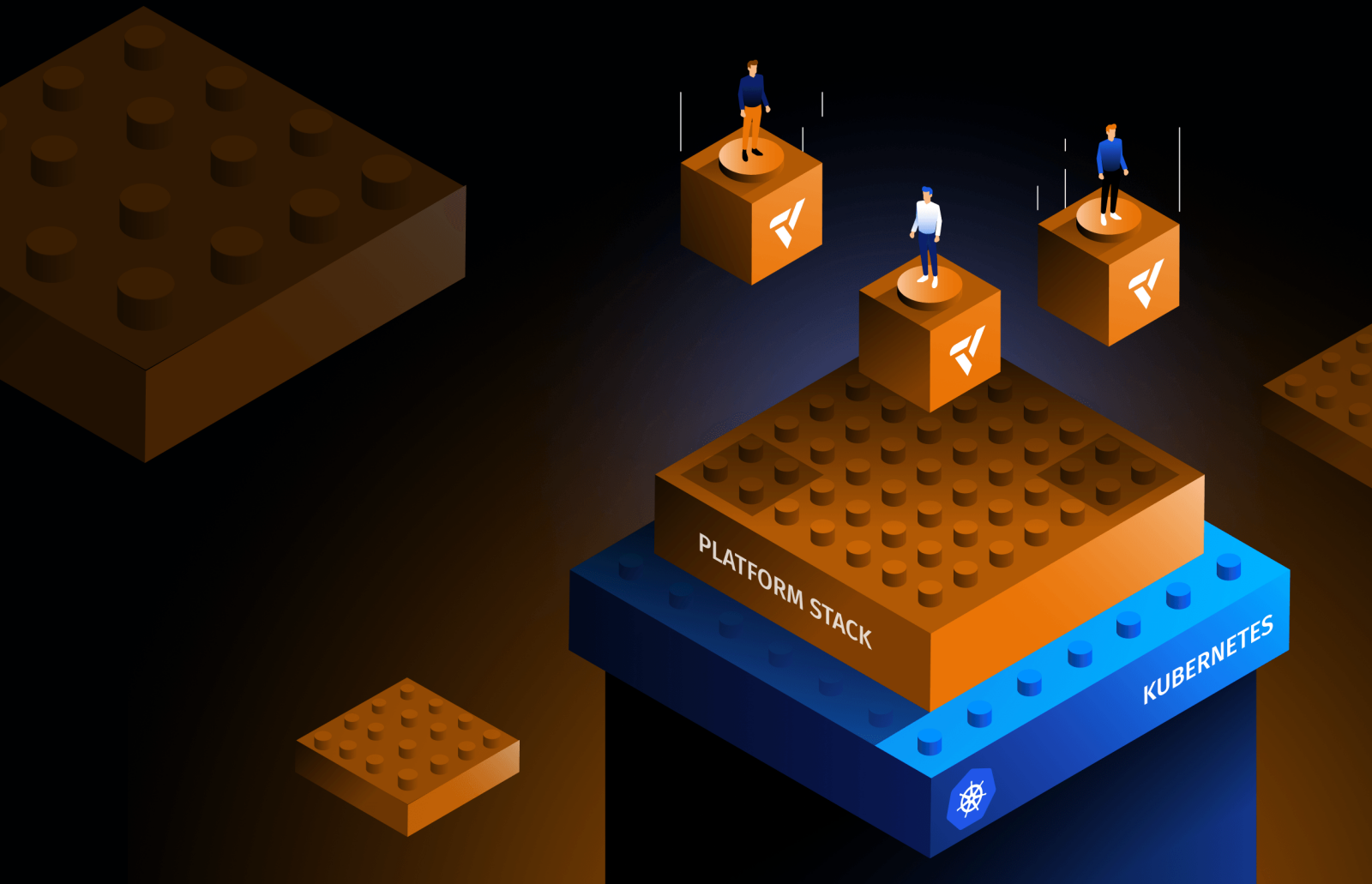


TABLE OF CONTENT

Introduction	3
Chapter 1: Mastering Kubernetes Cost Management: 7 Tips for Optimization	4
Cost Monitoring.....	4
Resource Limits.....	5
Autoscaling	5
Discounted Computing Resources	6
Sleep Mode	7
Cleanup.....	8
Cluster Sharing.....	9
Chapter 2: How to Save Kubernetes cost by reducing the number of clusters?	10
Advantages of fewer clusters.....	10
Disadvantages of fewer clusters	11
Chapter 3: Trade-Off: Efficiency vs. Stability	12
Chapter 4: How to Safely Reduce The Number of Required Clusters.....	13
Make Differentiated Decisions.....	13
Use Virtual Clusters	14
Implement Effective Multi-Tenancy	15
Chapter 5: How can Virtual Kubernetes clusters decrease costs?	16
An Example Scenario.....	17
Chapter 6: Switching to virtual Kubernetes clusters.....	19
conclusion	21

INTRODUCTION

Running Kubernetes can be very expensive, especially when it is done inefficiently. This is often the case when companies have just started to roll out Kubernetes in their organizations as then the same configuration and setup are often used that worked well for initial test projects or small applications. Such an initial unoptimized setup is probably normal. However, companies should also start to consider costs soon because this can save a lot of money and prevents the spread of bad practices.



CHAPTER 1

MASTERING KUBERNETES COST MANAGEMENT: 7 TIPS FOR OPTIMIZATION

Let's look at some ways to control and reduce your Kubernetes cost that can be applied to various Kubernetes use cases, from development and CI/CD to production.

Cost Monitoring

The first step to efficiently reduce your Kubernetes cost is to monitor these costs. This will not only help you to determine how much you saved eventually but also gives you an overview of what the most important cost drivers are.

Basic monitoring for your computing cost is provided by the public cloud providers as they are the ones ultimately charging you. Besides the total cost, you can also see in their billing overviews what exactly is running and what drives your cost, e.g. what proportion of your cost is attributed to computing, storage, and network traffic.

However, the overview of the cloud providers can only give you a basic understanding that is only limitedly helpful for multi-tenant Kubernetes clusters and of course is not available in private clouds. Therefore, it often makes sense to use additional tools to measure your Kubernetes usage and costs. Some useful tools in this area are Prometheus, Kubecost, and Replex.

Once you have set up your Kubernetes cost monitoring and identified areas of improvement, you can start with the actual cost optimizations.



Resource Limits

A good first cost-saving step is to set limits for resource usage. Efficient resource limits ensure that no application or user of your Kubernetes system consumes excessive computing resources. Therefore, they prevent you from unpleasant surprises in form of sudden cost increases.

Especially if you give developers direct Kubernetes access, e.g. in form of a self-service Kubernetes platform, limits are critical as they enforce a fair sharing of available resources, which keeps the total cluster size smaller. Without limits, one user could consume all resources leaving others unable to work, which will make them require more computing resources in total.

This goes not only for engineers in a Kubernetes multi-tenancy environment but also for applications running on the same cluster. In any case, it is of course important to set the right limits: If they are too low, engineers and applications cannot work properly and if they are too high, they are mostly useless. Here, the previously mentioned Kubernetes monitoring tools can help you to determine the right limits for the different use cases.

To implement resource limits, you can configure them Kubernetes-natively with Resource Quotas and Limit Ranges.

Autoscaling

Now that you have cost monitoring in place and prevented unexpected cost explosions, you can focus on saving costs. The best way to do this is to only pay for what you actually need. For this, it is important to get the size of your clusters, virtual clusters (vClusters), namespaces, etc. right.

Again, monitoring tools are helpful for this but manually observing and adjusting computing resources is not a very fast and flexible solution. To be able to respond to short-term fluctuations, you should therefore enable Kubernetes autoscaling.

Here, there are two types of autoscaling: Horizontal and vertical autoscaling. In short, horizontal autoscaling means adding and removing pods if the load is above or below a pre-defined threshold. With vertical autoscaling, the size of the individual pods is adjusted.

Both types of autoscaling are helpful to adjust your available computing resources automatically to your actual needs. However, this process is not always optimal because it may be too “conservative” or does not work for all use cases, e.g. there might be something running (requiring computing resources and thus not automatically downscaled) that is not used anymore. Therefore, some other measures should be implemented that will further drive your Kubernetes cost down.

Discounted Computing Resources

Another way to save computing cost is to use resources that have a discounted price. Of course, this only works in public clouds that offer such resources. However, all major cloud providers have some options for this:

AWS Spot Instances, GCP Preemptible VMs, and Azure Spot VMs provide heavily discounted computing resources that are “excess capacity” of the cloud providers. The providers sell this computing capacity cheaper as it would otherwise be unused. However, this also means that prices for these computing resources fluctuate depending on the demand and that your instance and application may be stopped if the price exceeds your limit or if there is no capacity available anymore. For this, spot instances are mostly an option for applications and workloads that are not needed permanently, e.g. if you want to run a computing-intense experiment. However, if you can use this type of discounted computing, you can save a lot of costs - AWS claims that you may get “up to a 90% discount”.

If spot instances and preemptible VMs are not an option for your application, for example, because you always need to run it without any disruption, you may also get a discount for committing to use the resources permanently for a specific time period. By committing to a one- or three-year usage period, you can get a significant discount - 40%-60% according to AWS. Such a long-term discount is available for AWS and Azure as “Reserved Instances”, and for Google as “Committed Use Discount”. This type of discount can make sense if you have a predictable and continuous need for computing resources over a longer period of time.

Overall, it depends on your type of application and computing needs, which type of discount is applicable. Here, it is also possible to use both types for different parts of your application or for different use cases. Therefore, even if the pricing of all public cloud providers is quite complicated, it can certainly pay off to take a closer look at the available discounts there so you get the same computing resources just at a lower price.

Sleep Mode

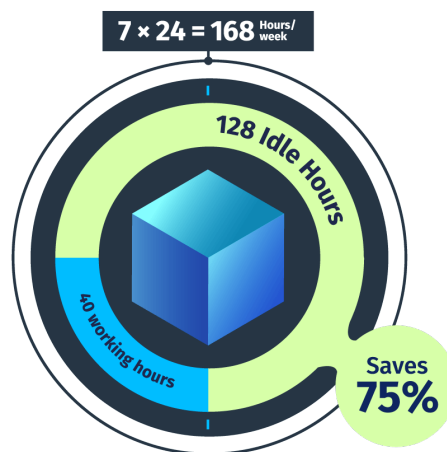
There are quite a few scenarios in which clusters, vClusters, and namespaces continue to run and create costs, although they are not needed anymore. These use cases often (but not only) happen when Kubernetes is used during development, testing, or CI/CD by engineers.

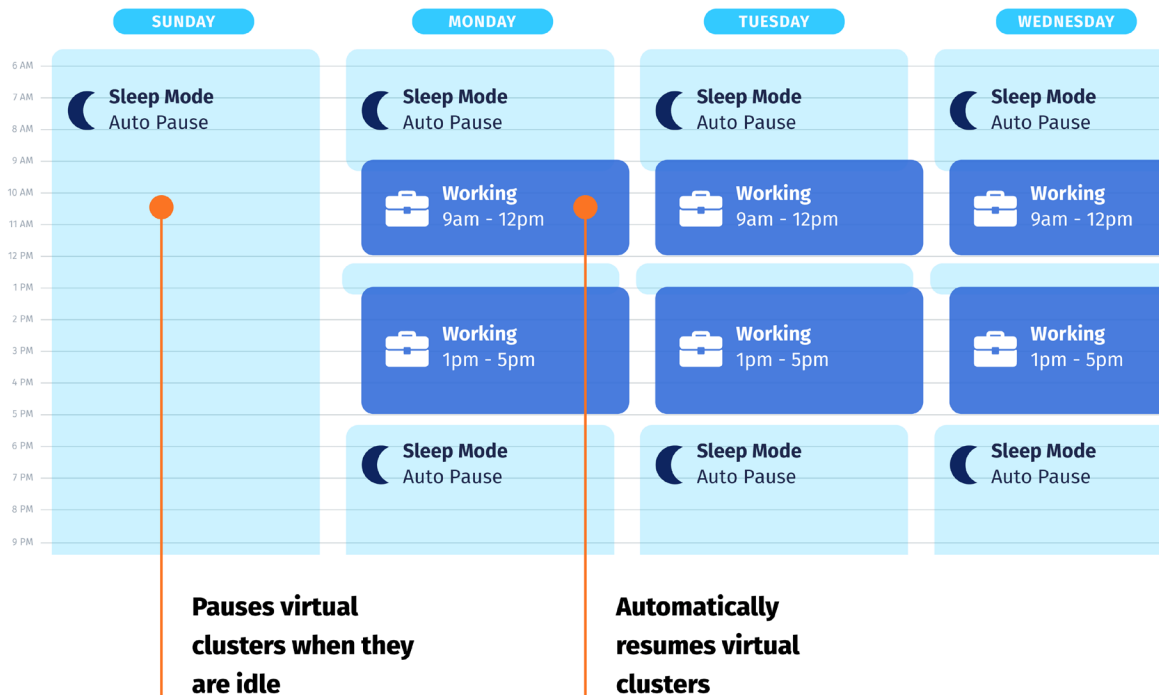
For example, if a developer is using a Kubernetes development environment in the cloud, they only need this environment during their work hours. If you assume that they work 40 hours per week with a Kubernetes environment (not accounting for meetings or holidays) and the environment is running all the time, you can save more than 75% (a week has 168 hours) by shutting off the environment when it is not needed.

The developers could of course shut down their environment themselves when they are done, but this is a process that is easily forgotten or ignored. It thus makes sense to automate it with a “sleep mode”, i.e. an automated process that scales down unused namespaces and virtual clusters. This ensures that the state of the environment is stored and that the environment can “wake up” again very fast and automatically when it is needed again so that the engineer’s workflow is not interrupted.

It is possible to implement such a sleep mode with scripts or by using tools that have a sleep mode inbuilt, such as Loft.

For a more detailed description and calculation of the cost-saving potential of a sleep mode, you should look at my article about saving Kubernetes cost for engineers.





Cleanup

Related to a sleep mode to scale down computing resources that are temporarily not needed, you should also clean up your Kubernetes system from time to time. Especially, if you allow engineers to create namespaces on demand or if you use Kubernetes for CI/CD, you may have a lot of unused namespaces or even clusters that still cost you money.

Even if you have a sleep mode in place that scales down the computing resources, the sleep mode is only intended for temporarily unused computing resources and thus preserves the state, including storage and configuration. However, especially to run CI/CD pipelines or tests, it is usually not necessary to preserve the state of the Kubernetes environment, so it is better to also delete it.

Again, this can be done with individual scripts or by using Loft, which provides such an auto-delete feature out-of-the-box. As an additional positive side-effect, deleting some environments makes it easier for admins to oversee the system, which can also be seen as an indirect cost-saving factor.

Cluster Sharing

A final Kubernetes cost-saving approach is to reduce the number of clusters. This generally saves cost because the control plane and the computing resources can be shared by several users or applications in a Kubernetes multi-tenancy setup. If you use a public cloud that charges a cluster management fee, such as AWS or Google Cloud, you can also save this fee. Finally, managing many clusters is a complex task, so limiting the number of clusters also relieves the admins.

Especially during the pre-production stage, many companies have too many clusters, sometimes even for every individual developer. This is clearly inefficient as a whole cluster is rarely needed during development and even if developers want to experiment with cluster-wide configurations, they could use a virtual cluster as dev environment instead. Therefore, relying on a multi-tenant internal Kubernetes platform for all non-production use cases can save a lot of cost by eliminating unnecessary redundancies.

However, also during production, reducing the number of clusters can make sense with the same underlying idea of improving resource utilization, reducing redundancies, and avoiding cluster management fees. Of course, this does not mean that you should only have one cluster. You should rather generally evaluate what user groups and applications can share a cluster and when dedicated clusters make sense.

CHAPTER 2

HOW TO SAVE KUBERNETES COST BY REDUCING THE NUMBER OF CLUSTERS?

When you are using Kubernetes at a larger scale and at different stages (development, testing, production), you will sooner or later face the question of how many clusters you should run. To find the right answer to this question is not easy as having many clusters has advantages and disadvantages compared to running only one or a few clusters, as discussed in this article.

However, your answer to this question also determines how much you will have to pay for your Kubernetes system. In general, it is cheaper to run only a few clusters, which is why I will explain in this article how you can reduce the number of clusters and thus save Kubernetes cost without negatively impacting your system.

Advantages of fewer clusters

No Redundancies: Running many clusters means that you also have many API servers, etcds and other parts of the control plane. This leads to a lot of redundancy and inefficiency as most of these components will be underutilized. However, if you only run a few clusters, all applications and users of the same cluster can share a control plane, which will drive utilization up and cost down significantly.

No Cluster Management Fees: Some public cloud providers, including AWS and Google Cloud, charge their users a flat cluster management fee for every cluster. Naturally, if you



have fewer clusters, you will pay less for the cluster management. The cluster management fee is particularly important for situations with many small clusters as the relative cost (about \$70 per month per cluster) can represent more than 50% of the total cost here.

Efficient Administration: In general, it is easier to manage and supervise a system with a limited number of clusters because you can get a much better overview of the system and do not have to repeat the same processes many times (e.g. updating every cluster). For this, reducing the number of clusters also reduces the admin effort for the clusters and will ultimately lead to additional cost savings.

Cluster sharing, which leads to fewer clusters, is also the basis for an internal Kubernetes platform that provides the engineers with self-service namespaces improving the Kubernetes development workflow. This illustrates that a reduction of the number of clusters can also be a positive side-effect of improvements on other parts of your Kubernetes system.

Disadvantages of fewer clusters

Single Point of Failure: If more applications are running or more users are working on the same cluster, all of them will be impacted at the same time when the cluster crashes. This means that the stability of the cluster will be more important as the cluster becomes a “single” point of failure. Of course, “single” in single point of failure does not necessarily mean that there is only one cluster but the same goes analogously for a low number of clusters.

Multi-Tenancy: If a cluster is shared by several users and applications, you need to implement an efficient Kubernetes multi-tenancy. This can be quite challenging because it does not only comprise the isolation of tenants but also user management and enforcement of limits. Especially hard multi-tenancy, i.e. a system that securely isolates tenants that are not trusted, is hard to establish with Kubernetes as some components are always shared within one cluster.

Technical Limitations: While a single Kubernetes cluster is quite scalable, its scalability will be limited at some point. For example, Kubernetes v1.19 supports clusters with up to 5,000 nodes, 150,000 total pods, and 300,000 total containers. These limits are of course pretty high, but you might already encounter scalability problems before reaching the limits, e.g. due to networking capacity. Therefore, it is also technically infeasible to run everything on just a single cluster if you reach a very large scale.

CHAPTER 3

TRADE-OFF: EFFICIENCY VS. STABILITY

Simply put, in determining the right number of clusters for your use case, you face a trade-off between (cost-)efficiency and stability of your system. Therefore, it is clear that your goal should not be to end up with just a single cluster and try to run everything there. To efficiently save Kubernetes cost, you should rather use the lowest reasonably possible number of clusters that still ensures the stability of your system. And this “right” number heavily depends on the size of your system, the number of users, and the nature of your software.

However, there are some approaches that can help you to drive down the number of required clusters and so will save you cost:

Reducing the number of clusters is just one of several methods to save costs. In my guide to reducing Kubernetes cost, I also explain some other approaches that make your Kubernetes system more cost-efficient.



CHAPTER 4

HOW TO SAFELY REDUCE THE NUMBER OF REQUIRED CLUSTERS

Make Differentiated Decisions

Since there is no one-size-fits-all answer to the questions of how many clusters you should run, you need to make an individual decision for your use case. However, it is not enough to make this decision once, but you should rather repeat it for every part of your application and for every group of users.

This means that you should not just adopt a dogmatic policy of “every engineer and every application get an own cluster” or “we are doing everything in one cluster”. You really need to evaluate if it makes sense in the situation at hand.

For example, there may be some applications, especially very critical applications running in production, that should get their own cluster so that these applications cannot be impacted by others. Other applications may share a cluster either because they are less important or are only used for developing and testing, which makes them less critical. However, also critical applications may run on a shared cluster, e.g. if they heavily depend on each other and would not work anyway if one of them failed.

You also need to make the cluster decision for your engineers: Usually, not every engineer needs an own cluster but namespaces are often enough, so you might provide them with self-service namespaces that run on just one cluster. Still, some engineers may have special

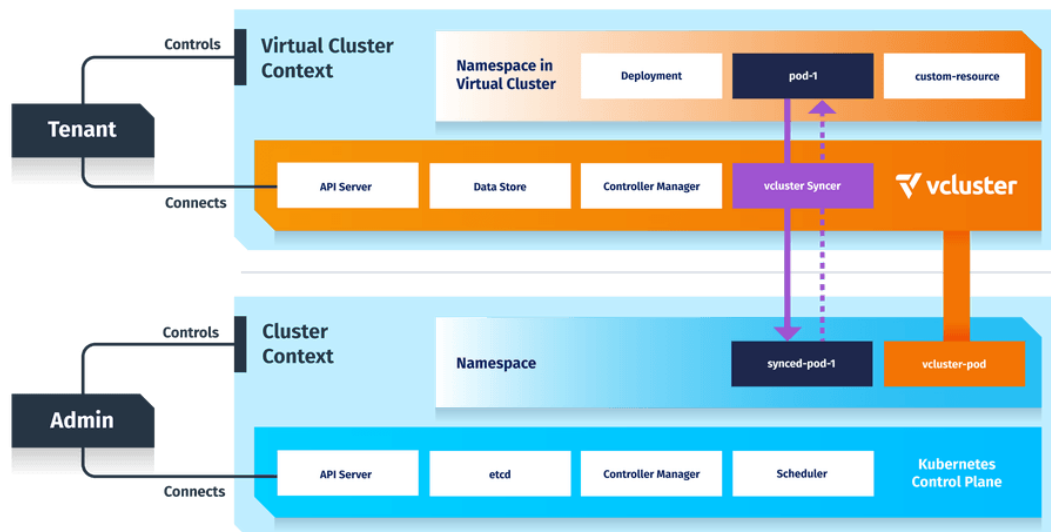


requirements and need to work directly on the Kubernetes configuration. Since this is not possible with simple namespaces, these engineers should get individual clusters or alternatively virtual clusters, which I will describe next.

If you want to learn more about the decision about clusters for development purposes, take a look at this comparison of individual clusters and shared clusters for development.

Use Virtual Clusters

Another way to decrease the number of clusters is to use virtual clusters (vClusters). Virtual clusters are very similar to “real” clusters but are virtualized and several virtual clusters can thus run on one physical cluster. Virtual clusters have some advantages that allow you to replace physical clusters with vClusters:



Virtual clusters solve the multi-tenancy issue as they provide better isolation of tenants than namespaces, which is why it is easier to implement hard multi-tenancy with vClusters. Additionally, the tenants can configure their vCluster freely and independently from others, so that vClusters are very good development environments even for engineers that need to configure Kubernetes or use a specific version of it. Finally, you can use vClusters for cluster sharding, so it is, for example, possible to handle more requests and thus push the feasible technical limits of Kubernetes.

Since virtual clusters are still running on a single physical cluster, this physical cluster remains a single point of failure. However, as the configurations and additional installations can be done on the vCluster level, the underlying cluster can be rather simple with only basic components, which makes this cluster less error-prone and thus more stable.

Implement Effective Multi-Tenancy

One of the reasons why some companies prefer to use many clusters is that they are not sure how to implement multi-tenancy and how much effort this is. If more organizations get best practices for Kubernetes multi-tenancy right, they could reduce the number of clusters, especially for non-production use cases.

Fortunately, Kubernetes has some helpful features for multi-tenancy, such as Role-based access control (RBAC) and Resource Quotas. In general, setting up a good multi-tenant system is mostly a one-time effort and requires good user management and strictly enforced user limits. To get this, some companies such as Spotify started to build their own internal Kubernetes platforms, but it is also possible to buy out-of-the-box solutions for this, such as Loft.

Reducing the number of clusters can lead to significant cost savings. However, just using a single cluster is often not the right solution as this could negatively impact the stability of your system. Finding the right number of clusters for your use case is therefore not an easy task because there is no general rule of how many clusters are optimal.

It is rather a question you have to answer for every application and every group of engineers individually by assessing their situation and needs. However, it is possible to reduce the minimum number of necessary clusters by replacing some clusters with virtual clusters and by implementing efficient multi-tenancy.

This will allow you to improve the cost-efficiency of your Kubernetes system without a negative impact on its stability.

CHAPTER 5

HOW CAN VIRTUAL KUBERNETES CLUSTERS DECREASE COSTS?

Virtual Kubernetes clusters are a trade-off between namespaces and separate Kubernetes clusters. They are easier and cheaper to create than fully blown clusters. They provide much better isolation than namespaces. Virtual clusters use a completely separate control plane, and within a virtual cluster you have full cluster-wide control access.

Virtual Kubernetes clusters provide an excellent opportunity to replace separate clusters and drastically reduce your infrastructure and management costs, especially in scenarios where you have at least basic trust in your tenants (say separate teams across your company, CI/CD pipelines, or even several trusted customers). You can check out this [blog post](#) and learn about Kubernetes cost optimization.

Virtual Kubernetes clusters are fully functional Kubernetes clusters that run within another Kubernetes cluster. The difference between a regular Kubernetes namespace and a virtual cluster is that a virtual cluster has its own separate Kubernetes control plane and storage backend. Only a handful of core resources, such as pods and services, are actually shared among the virtual and host cluster. All other resources, such as CRDs, statefulsets, deployments, webhooks, jobs, etc., only exist in the pure virtual Kubernetes cluster.

This provides a lot better isolation than a regular Kubernetes namespace and decreases the pressure on the host Kubernetes cluster as API requests to the virtual Kubernetes cluster in most cases do not reach the host cluster at all. In addition, all created resources by the virtual cluster are also tied to a single namespace in the host cluster, no matter in which virtual cluster namespace you create those resources in.



The table below summarizes the differences between Namespaces, virtual Kubernetes clusters, and fully separate clusters.

	Separate Namespace For Each Tenant	 vCluster	Separate Cluster For Each Tenant
Isolation	very weak	strong	very strong
Access For Tenants	very restricted	vcluster admin	cluster admin
Cost	very cheap	cheap	expensive
Resource Sharing	easy	easy	very hard
Overhead	very low	very low	very high

Virtual clusters compared with namespaces and traditional clusters

The important takeaway from this is that virtual Kubernetes clusters provide a new alternative to both namespaces and separate clusters. Virtual Kubernetes clusters provide an excellent opportunity to replace separate clusters and drastically reduce your infrastructure and management costs, especially in scenarios where you have at least basic trust in your tenants (say separate teams across your company, CI/CD pipelines, or even several trusted customers).

An Example Scenario

Let's say you are a company that provides some sort of SaaS service, and you have around 100 developers distributed across 20 teams that implement different parts of the service. For each of those 20 teams, you provisioned separate Kubernetes clusters to test and develop the application, as this was the easiest and most flexible approach. Each team's cluster has at least three nodes to guarantee availability and then automatically scales up and down based on usage.

Your minimum infrastructure bill in Google Cloud might look like this over 12 months (according to the Google Cloud Pricing Calculator):

Node Cost: 20 Clusters 3 Nodes (n1-standard-1) = $20 * 3 * \$72.82 = \$4,369.20$

GKE Management Cost: 20 Clusters (Zonal) = $20 * \$71.60 = \$1,432.00$

Total Cost Per Year (Without Traffic etc.): $\$4,369.20 + \$1,432.00 = \$5,801.20$

In total, you are looking at a minimum estimated raw node + management cost of about \$5,800. Obviously, you could still fine-tune certain aspects here, for example reducing the minimum node pool size or using preemptive nodes instead of regular nodes.

The advantages of this setup are clear. Each team has its own separate Kubernetes cluster and endpoint to work with and can install cluster-wide resources and dependencies (such as a custom service-mesh, ingress controller, or monitoring solution). On the other hand, you'll also notice that the cost is quite high. Resource sharing across teams is rather difficult, and there is a huge overhead if certain clusters are not used at all.

CHAPTER 6

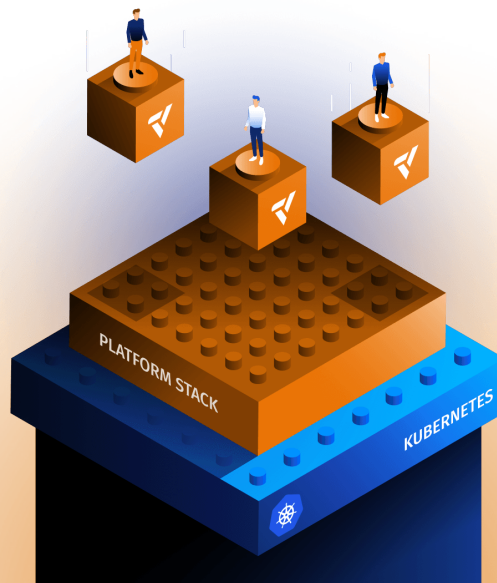
SWITCHING TO VIRTUAL KUBERNETES CLUSTERS

This is a perfect example where virtual Kubernetes clusters could come in handy. Instead of 20 different GKE clusters, you would create a single GKE Kubernetes cluster and then deploy 20 virtual Kubernetes clusters within it. Now each team gets access to only a single virtual Kubernetes cluster endpoint that essentially maps to a single namespace in the underlying GKE cluster.

The really great part about this is that from the developers' perspective, nothing has changed. Each team can still create all the cluster services they want within their own virtual cluster, such as deploy their own Istio service mesh, custom cert-manager version, Prometheus stack, Kafka operator, etc. without affecting the host cluster services. They can essentially use it the same way as they would have used the separate cluster before.

Another benefit is that the setup is now much more resource-efficient. Since the virtual Kubernetes clusters and all of their workloads are also just simple pods in the host GKE cluster, you can leverage the full power of the Kubernetes scheduler. So, for example, if a team is on vacation or is not using the virtual Kubernetes cluster at all, there will be no pods scheduled in the host cluster consuming any resources. In general, this means the node resource utilization of the GKE cluster should now be much better than before.

Another significant advantage with a single GKE cluster and multiple virtual clusters in it is that you as the infrastructure team can centralize certain services in the host cluster, such as a central ingress controller, service mesh, metrics, or logging solutions instead of installing it every time into all of the separate clusters. The virtual clusters will be able to consume those services, or if the teams prefer they can still add their own. Teams will also be able to access each other's services if that is needed, which would be very difficult with completely separate clusters.



Furthermore, you will save on the cloud provider Kubernetes management fees by sharing resources better. And vCluster is open source, so it's self-managed and completely free. The new cost estimate would look a little bit more like this if you would reserve a node for each team and add three extra nodes as a high availability buffer:

Node Cost: 1 Clusters 23 Nodes (n1-standard-1) = 12 \$558.26 = \$6699.12

GKE Management Cost: 1 Clusters (Zonal) = 12 1 \$71.60 = \$859.20

Total Cost Per Year (Without Traffic etc.): \$6699.12 + \$859.2 = \$7558.32

Total Cost Savings: \$34,660.80 - \$7558.32 = \$27,102.48 (78.2% savings)

In this case, your minimum node & GKE management fee infrastructure bill would be cut down by 78.2%. This is a rather constructed example, but it shows the considerable potential of virtual clusters. For the teams using the virtual clusters, essentially nothing would change because they would still have access to a fully functional Kubernetes cluster where they could deploy their workloads and cluster services freely.

Virtual Kubernetes clusters are a third option if you have to decide between namespaces or separate clusters. Virtual clusters will probably never completely replace the need for separate clusters. Still, they have significant advantages if your use case fits, as you can save significant infrastructure and management costs with them.



CONCLUSION

The cost of running Kubernetes at a larger scale and with many users is an issue.

The first step to manage your Kubernetes cost is to get an overview and to start monitoring the cost. Then, you should implement limits to prevent excessive computing resource consumption, which makes your cost more predictable. To reduce your cost, finding the right size of your clusters, virtual clusters, and namespaces is important and autoscaling can be very helpful for this. You should also take a look at the discount options of public cloud providers if they are an option for you. An automatic sleep mode and cleanup for unused Kubernetes namespaces and vClusters are further measures to eliminate idle resources. Finally, you should consider reducing the number of clusters because this improves resource utilization and reduces unnecessary redundancies, especially at the pre-production stage.

If you implement all these measures, your Kubernetes system is quite cost-optimized and thus much cheaper than any initial best guess setup. Then, you should be ready to tackle your next challenges in using Kubernetes without having to fear its cost.





Go To [VCLUSTER.COM](https://vcluster.com) To Get Started Today Or
Email CONTACT@LOFT.SH To Schedule a Demo

