How to Optimize Kubernetes Autoscaler to Better Business

Kubernetes, the open-source tool, is one of the most popular containers. Initially conceptualized and owned by Google, Kubernetes was made available for free. While other containers such as docker and vagrant are widely used, Kubernetes has garnered enough traction to be in its league. This is primarily due to how easy, and cost-effective Kubernetes or Kubes is.

While Kubernetes, also known as K8s, is popular among developers, not every developer is aware of the true potential of the container. So, while Kubernetes is used for its easy development and deployment process, Kubernetes is also in demand for its Autoscaling feature.

Kubernetes Autoscaling is a key feature that allows seamless scaling with multiple mechanisms. While the cloud has made scaling synonymous with cakewalk, there is

Headquarters: Veritis Group, Inc , 1231 Greenway Drive, Suite 1040, Irving, TX 75038

Phone: 972-753-0022 | **Email:** connect@veritis.com

much more to the scaling than one assumes, and Kubernetes made the scaling process better than most.

As alluded to, Kubernetes has different scaling methods, which we shall explore in this blog. For the curious soul, the three methods of Kubernetes autoscaling are Horizontal Pod Autoscaler, Vertical Pod Autoscaler, and Cluster Autoscaler. We shall look into these processes and how they help the automation of application scaling.

Understanding Kubernetes Autoscaling

Automation is an aspect that companies are still finding their way about. The shrewdest ones embrace Kubernetes as one can automate processes with Kubes, ranging from provisioning to scaling. This automation saves time and money and wicks away manual intervention as automated scaling or autoscaling handles all these issues, ranging from demand spikes to scaling down resources. This method can be used along with Cluster Autoscaler, which, like Kubernetes autoscaling, uses the required resources without intervention.

While ‘autoscaling’ bears the semblance of a ‘good-to-have feature,’ it is a must-have should you keep experiencing spikes and have a lot of error-prone tasks. You dedicate resources and time to make the required changes in manual scaling.

But, with autoscaling, you would not have to intervene as resources are either roped in or shed as and when required. With this dynamic and responsive scaling, Kubernetes Autoscaling saves your money and improves your quality. While it sounds like there is one autoscaling, there are three different autoscaling methods in Kubes.

Let’s look at them!

Kubernetes Horizontal Pod Autoscaler

The ‘Horizontal’ in Kubernetes Horizontal Pod Autoscaler would appear familiar to the people familiar with horizontal scaling. The Kubes Horizontal Austoscaler modifies the number of pods to meet the growing demand. It effectively modifies the workload aspects, which may range from deployment to StatefulSet, and thereby it ensures that the company meets the demand it is required to.

The Kubes Horizontal Pod also has the diligence to shed the resources when the demand dips. This process allows the software applications to scale up or down without manual intervention. The scale-down may not appear important, but when a software application scales down, it frees up the nodes which are no longer fully utilized. Other resources will be saved by shedding the unwanted nodes, and you will be laden with a cost-effective Kubernetes solution.

Headquarters: Veritis Group, Inc , 1231 Greenway Drive, Suite 1040, Irving, TX 75038

Phone: 972-753-0022 | **Email:** connect@veritis.com

Also, Kubernetes Horizontal Pod Autoscaling doesn't gel with unscalable loads. This includes DaemonSets, which perform on the "one pod per node" principle. These unscalable loads remove the chance of replicas and thereby remove the possibility of executing horizontal pod autoscaling.

Useful Link: Top 10 DevOps Tools to Pick for Your Business

Kubernetes Vertical Pod Autoscaler

Kubes Vertical Pod Autoscaler or VPA is a tool that allows you to boost up the existing pods in terms of processing and memory. This method can be embraced with one required to rope in more processing power to get the task done, and the VPA scales up the concerned values and makes it up-to-date.

This scaling-up includes memory and CPU power and keeps a check on the number of containers in the pods. As Kubernetes VPA scales up the power of the required pods, it also frees up CPU and memory resources in other pods.

Some of the benefits of VPA is that your pods will become quite efficient as they will use only the required number of nodes. You would also save time as VPA would take care of

the mundane benchmarking procedures, typically about identifying the optimal CPU capacity and other aspects. With all these aspects taken care of, your maintenance time shall dip as VPA maintains everything automatically.

Kubernetes Cluster Autoscaling

The Kubernetes Autoscaler adds or removes nodes to raise or reduce the size of a Kubernetes cluster based on the availability of pending pods and node utilization metrics.

The Kubernetes Autoscaler adds nodes to the cluster when it detects waiting for pods that can't be scheduled, possibly causing resource restrictions. Furthermore, when a node's usage falls below a certain threshold established by the cluster administrator, the node is removed from the cluster, enabling all pods to function and removing unneeded nodes.

Delving Deeper into Nodes

EKS is in charge of Kubernetes-as-a-Service. This, however, largely applies to the Kubernetes control plane. For the pods to be deployed, you must still contribute compute nodes to your EKS Cluster. You may build worker nodes for your EKS Cluster in two methods.

- **Self-Managed Nodes**

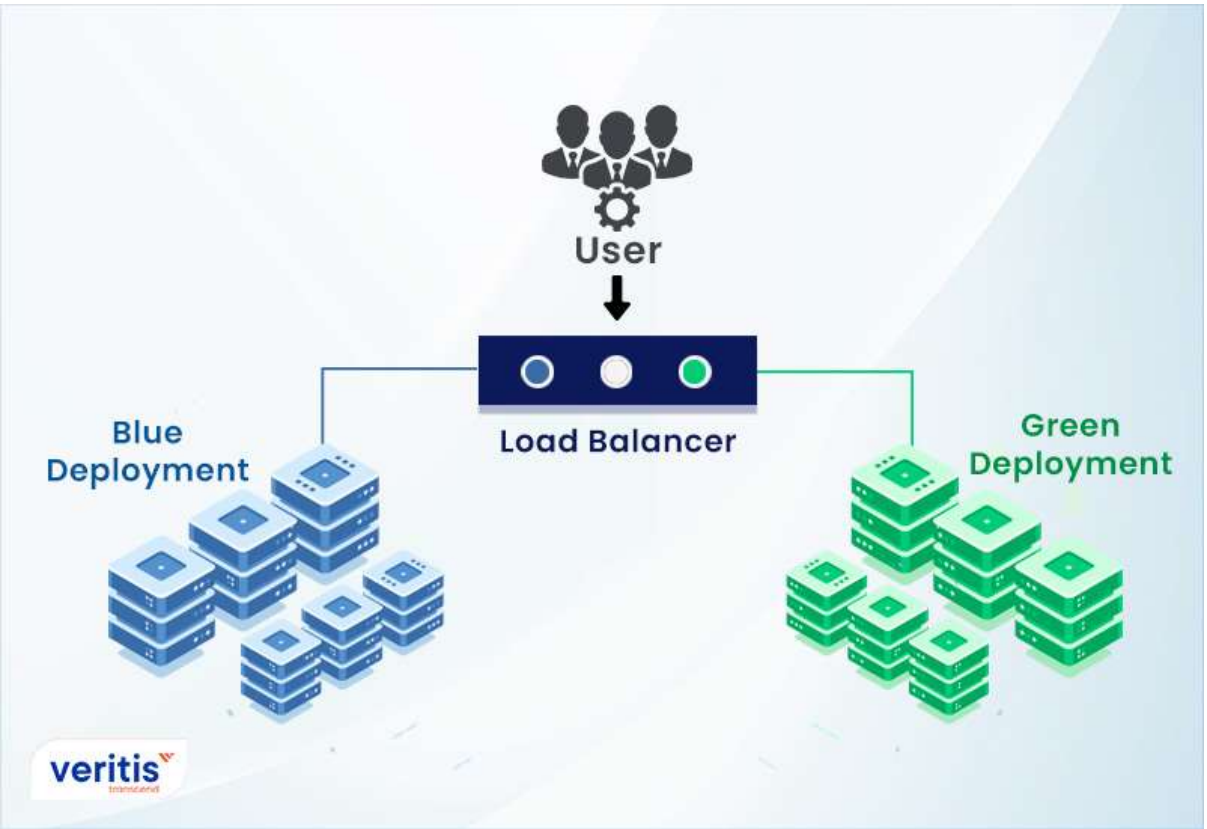
You may establish, maintain, and administer your worker nodes using this tool, giving you more authority over your platforms.

- **EKS Managed Nodes**

You don't have to manage the nodes in the EKS Kubernetes Cluster if you choose that option. However, you have to define specifications, and EKS will handle everything else.

Useful Link: **EKS Vs. AKS Vs. GKE: Which is the right Kubernetes platform for you?**

Understanding Blue-Green Deployment and Rolling Updates in Kubes



Amongst the most crucial things to understand is how to safely roll out the new version of the program while the previous one is still running. Can the existing pods be replaced without causing traffic congestion? That is one of the biggest questions one would have while dealing with Kubes and its autoscaling feature.

Kubernetes can be deployed in different ways. We've provided two choices below that allow for secure deployment while keeping the possibility to go back in time if needed.

Blue-Green

Blue-green deployment is a method that involves running two identical production environments, Blue and Green, to minimize outages and risks. Either of the environments is active at any given moment, with the active instance providing all operational data. Blue is currently active, while Green is inactive in this case.

Deployment and the final round of testing for a new version of your program occur in a non-live environment, such as Green in this case. You change the router so that all incoming queries flow to Green instead of Blue once you've deployed and completely tested the software in Green. Green is now active, whereas Blue is dormant.

Rolling Update

One of the key benefits of utilizing a Deployment to regulate your pods is the ability to execute rolling updates. Deployments provide you with complete control over the process while rolling updates let you gradually alter the configuration of your pods. When configuring rolling updates, update planning is the main aspect.

In the Deployment declaration, your strategy will have two possible values:

- Rolling Update: New pods are gradually introduced, while old pods get slowly phased out in this rolling update.
- Recreate: The old pods are ended well before fresh pods get placed.

While Kubernetes autoscaling is great, it is always good to explore other options as it would help you use another tool should one give away.

Karpenter

Karpenter is a Kubernetes Cluster Autoscaler built on AWS that is open-source, configurable, and high-performing. It boosts application availability and cluster efficiency by rapidly deploying the right computing resources in response to changing application loads.

Karpenter also provides ad hoc computing resources to meet your application's needs, and it will soon provide a cluster compute resource footprint, which will reduce costs while improving performance. In addition, Karpenter analyses the overall resource requirements of unplanned pods and chooses whether to deploy new nodes or terminate current ones to decrease planning delays and capital costs.

Useful Link: Kubernetes Adoption: The Prime Drivers and Challenges

---

Kubernetes-based Event-Driven Autoscaling (KEDA)

Microsoft and Red Hat built KEDA (Kubernetes-based Event-driven Autoscaling), an open-source component that allows Kubernetes workloads to gain from an event-driven architecture approach.

A Kubernetes Deployment or Job may be horizontally scaled with KEDA. It is built on the Kubernetes Horizontal Pod Autoscaler. It allows users to provide autoscaling criteria based on data from any event source, such as Kafka topic lag with Prometheus query parameters. When autoscaling a Deployment or a Job, you may choose from a list of pre-defined triggers that serve as a source of events and statistics.

Capping it off

As we end the blog here, we believe you have walked away with more than a general idea of how Kubernetes Autoscaling would assist you in scaling resources in and out of your Kubernetes Cluster. Also, keep in mind that the Kubernetes Autoscaling layers (nodes/pods) are vital and interconnected.

HPA may be used to scale the number of pods coming in and out. For instance, raise the pod count as consumption rises and decrease the pod count as user load falls. To assist you in “right-size” your applications, HPA raises or lowers the pod count, whereas VPA automatically increases or decreases the CPU and memory reservations of the pods.

HPA and VPA accomplish pod-level Kubernetes autoscaling. To expand the number of nodes in the cluster, you’ll require the Kubernetes Autoscaler. This ultimately brings Kubes back into the picture.

While Kubernetes autoscaling sounds easy, using the containers isn’t as easy as one would assume. The experienced people who have dealt with it realized that the Kubes know there is more to the containers than meets the eye. The wise novices and experienced companies with heavy workloads rope in the services of Veritis.

Committed to excellence, Stevie Award Veritis has awed its clients with its amazing solutions. Be it an emerging company or a Fortune 500 organization; we have enough expertise to support your unique needs.

So, reach out to us and experience the **benefits of Kubernetes** autoscaling.

Services

Headquarters: Veritis Group, Inc , 1231 Greenway Drive, Suite 1040, Irving, TX 75038

Phone: 972-753-0022 | **Email:** connect@veritis.com