Cluster Autoscaler Compared in GKE, EKS, and AKS

In this lesson, we will see the comparison between Cluster Autoscaler in GKE, EKS and AKS.

WE'LL COVER THE FOLLOWING

Kubernetes's service providers

`Cluster Autoscaler` is a prime example of the differences between different managed Kubernetes offerings. We'll use it to compare the three major Kubernetes-as-a-Service providers.

> 🔍 I'll limit the comparison between the vendors only to the topics related to Cluster Autoscaling.

GKE service provider

GKE is a no-brainer for those who can use Google to host their clusters. It is the most mature and feature-rich platform. They started Google Kubernetes Engine (GKE) long before anyone else. When we combine their headstart with the fact that they are the major contributor to Kubernetes and hence have the most experience, it comes as no surprise that their offering is way above others.

Setting up `Cluster Autoscaler` in GKE

When using GKE, everything is baked into the cluster. That includes Cluster Autoscaler. We do not have to execute any additional commands. It simply works out of the box. Our cluster scales up and down without the need for our involvement, as long as we specify the `--enable-autoscaling` argument when creating the cluster. On top of that, GKE brings up new nodes and joins them to the cluster faster than the other providers. If there is a need to expand the cluster, new nodes are added within a minute.

There are many other reasons I would recommend GKE, but that's not the subject right now. Still, Cluster Autoscaling alone should be proof that GKE is the solution others are trying to follow.

EKS service provider

Amazon's Elastic Container Service for Kubernetes (EKS) is somewhere in the middle. Cluster Autoscaling works, but it's not baked in. It's as if Amazon did not think that scaling clusters is important and left it as an optional add-on.

EKS installation is too complicated (when compared to GKE and AKS) but thanks to eksctl from the folks from WeaveWorks, we have that, more or less, solved. Still, there is a lot left to be desired from eksctl. For example, we cannot use it to upgrade our clusters.

The reason I'm mentioning eksctl in the context of auto-scaling lies in the **Cluster Autoscaler** setup.

Setting up `Cluster Autoscaler` in EKS

I cannot say that setting up `Cluster Autoscaler` in EKS is hard. It's not. And yet, it's not as simple as it should be. We have to tag the Autoscaling Group, put additional privileges to the role, and install `Cluster Autoscaler`. That's not much. Still, those steps are much more complicated than they should be. We can compare it with GKE. Google understands that auto-scaling Kubernetes clusters is a must and it provides that with a single argument (or a checkbox if you prefer UIs). AWS, on the other hand, did not deem auto-scaling important enough to give us that much simplicity. On top of the unnecessary setup in EKS, the fact is that AWS added the internal pieces required for scaling only recently. `Metrics Server` can be used only since September 2018.

My suspicion is that AWS does not have the interest to make EKS great by itself and that they are saving the improvements for Fargate. If that's the case (we'll find that out soon), I'd characterize it as "sneaky business". Kubernetes has all the tools required for scaling Pod and nodes and they are designed to be extensible. The choice not to include **Cluster Autoscaler** as an integral part of their managed Kubernetes service is a big minus.

AKS service provider

What can I say about Azure Kubernetes Service (AKS)? I admire the improvements Microsoft made in Azure as well as their contributions to Kubernetes. They do recognize the need for a good managed Kubernetes offering. Yet, `Cluster Autoscaler` is still in beta. Sometimes, it works; more often than not, it doesn't. Even when it does work as it should, it is slow. Waiting for a new node to join the cluster is an exercise in patience.

Setting up `Cluster Autoscaler` in AKS

The steps required to install `Cluster Autoscaler` in AKS are sort of ridiculous. We are required to define a myriad of arguments that were supposed to be already available inside the cluster. It should know what is the name of the cluster, what is the resource group, and so on and so forth. And yet, it doesn't. At least, that's the case at the time of this writing. I hope that both the process and the experience will improve over time. For now, from the perspective of auto-scaling, AKS is at the tail of the pack.

Conclusion

You might argue that the complexity of the setup does not really matter. You'd be right. What matters is how reliable `Cluster Autoscaling` is and how fast it adds new nodes to the cluster. Still, the situation is the same. GKE leads in reliability and speed. EKS is the close second, while AKS is trailing behind.

> Q
>
> Microsoft understands that auto-scaling Kubernetes clusters is a must and it provides that with a single argument (or a checkbox if you prefer UIs)?

In the next lesson, we will revise and test the concepts of this second chapter, through a short quiz.