Autoscaling in Kubernetes

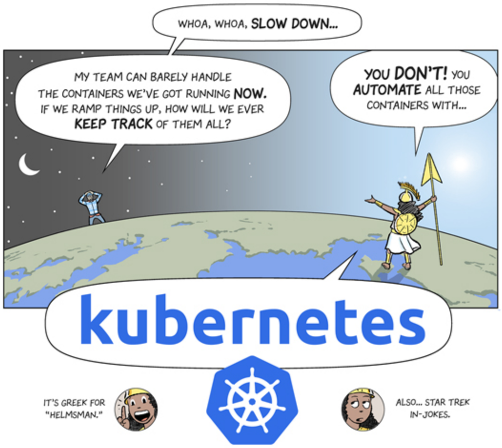
Abdullah Al Hasib

abdullah.alhasib@kratosdefense.com

09-02-2022

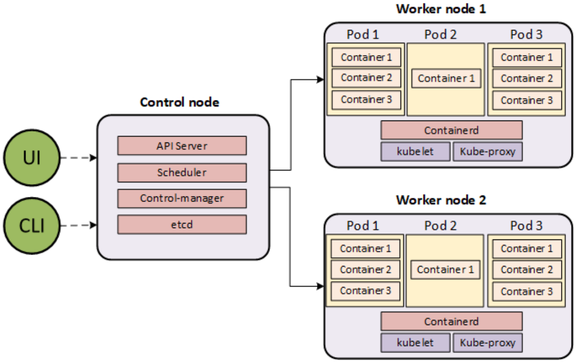
A Quick Recap

What is kubernetes:


WHOA, WHOA, SLOW DOWN...
MY TEAM CAN BARELY HANDLE THE CONTAINERS WE'VE GOT RUNNING NOW. IF WE RAMP THINGS UP, HOW WILL WE EVER KEEP TRACK OF THEM ALL?
YOU DON'T! YOU AUTOMATE ALL THOSE CONTAINERS WITH...
kubernetes
IT'S GREEK FOR "HELMSMAN."
ALSO... STAR TREK IN-JOKES.

A Quick Recap

Kubernetes architecture:



Autoscaling

Autoscaling:

- automatically adjusts computational resource usage according to the load.

Why do we need it:

- to cope with the demand
- to reduce cost
- to reduce power consumption

Autoscaling Types

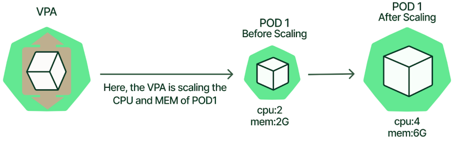
The Kubernetes autoscaling mechanism uses two layers:

- Pod-based scaling - supported by
  - Horizontal Pod Autoscaler (HPA)
  - Vertical Pod Autoscaler (VPA)
- Node-based scaling - supported by
  - Cluster Autoscaler (CA)

	Pods	**Nodes**
Horizontal	# of pods	# of nodes
Vertical	resources of a pod	resources of a node

Virtical Pod Autoscaling

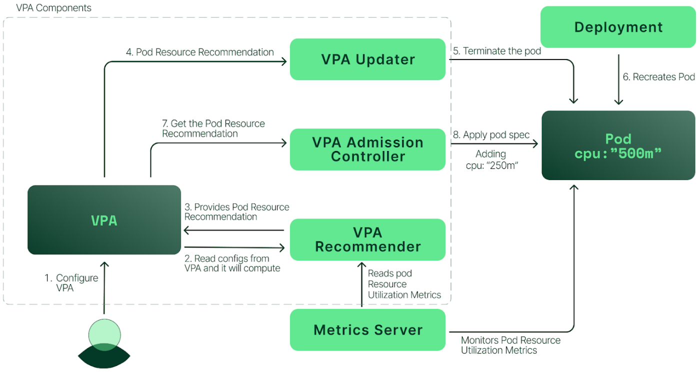
VPA adjusts the resource requests and limits of containers in the cluster.



Resource configuration types:

- requests: define the minimum amount of resources that containers need
- limits: define the maximum amount of resources that a given container can consume

How Does VPA Work?


VPA Components
4. Pod Resource Recommendation
VPA Updater
5. Terminate the pod
Deployment
6. Recreates Pod
7. Get the Pod Resource Recommendation
VPA Admission Controller
8. Apply pod spec
Adding cpu: "250m"
Pod cpu:"500m"
VPA
3. Provides Pod Resource Recommendation
VPA Recommender
2. Read configs from VPA and it will compute
1. Configure VPA
Reads pod Resource Utilization Metrics
Metrics Server
Monitors Pod Resource Utilization Metrics

VPA Concepts

Types Operation Modes:

- Off: default mode
- Auto: may cause service downtime
- Recreate: should be used rarely
- Initial

Types of metrics server APIs:

- Resource Metrics: Predefined resource usage metrics (CPU and memory)
- Custom Metrics: Custom metrics associated with a K8s object
- External Metrics: Custom metrics not associated with a K8s object

Limitations

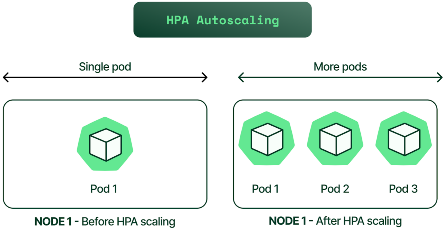
- Do not use VPA with HPA while they are using the same resource metrics such as CPU and memory usage.
- VPA might recommend more resources than available in the cluster, thus causing the pod to not be assigned to a node.
  - To overcome this limitation, set the LimitRange to the maximum available resources.

VPA Demo

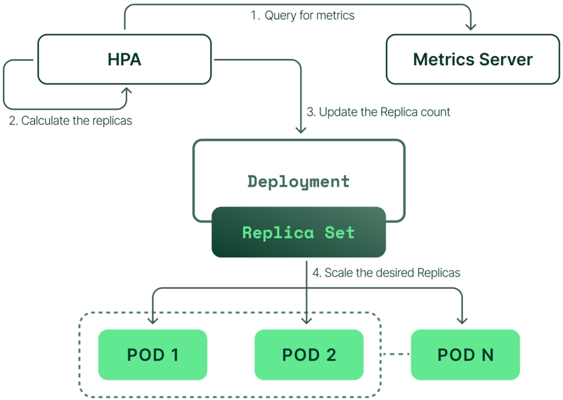
VPA Demo

Horizontal Pod Autoscaling

HPA increases or decreases the number of pods in a replication controller, deployment, replica set etc. based on CPU utilization



How Does HPA Work?


1. Query for metrics
HPA
Metrics Server
2. Calculate the replicas
3. Update the Replica count
Deployment
Replica Set
4. Scale the desired Replicas
POD 1
POD 2
POD N

Limitations

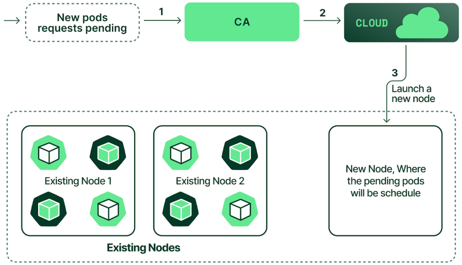
- One of HPA's most well-known limitations is that it does not work with DaemonSets.
- If you don't efficiently set CPU and memory limits on pods, your pods may terminate frequently or, on the other end of the spectrum, you'll waste resources.
- If the cluster is out of capacity, HPA can't scale up until new nodes are added to the cluster. Cluster Autoscaler (CA) can automate this process.

HPA Demo

HPA Demo

Cluster Autoscaling

CA adjusts the number of nodes in the cluster when pods fail to schedule or when nodes are underutilized


New pods requests pending
1
CA
2
CLOUD
3
Launch a new node
Existing Node 1
Existing Node 2
New Node, Where the pending pods will be schedule
Existing Nodes

Limitations

- CA does not make scaling decisions using CPU or memory usage. It only checks a pod’ s requests and limits for CPU and memory resources. This limitation means that the unused computing resources requested by users will not be detected by CA, resulting in a cluster with waste and low utilization efficiency.
- Whenever there is a request to scale up the cluster, CA issues a scale-up request to a cloud provider within 30–60 seconds. The actual time the cloud provider takes to create a node can be several minutes or more. This delay means that your application performance may be degraded while waiting for the extended cluster capacity.

Conclusion

- Here are two best practices for making efficient use of Cluster Autoscaler: Ensure resource availability for the Cluster Autoscaler pod—you can do that by defining a minimum of one CPU for resource requests made to the cluster autoscaler pod. It is critical to ensure that the node running the cluster autoscaler pod has enough resources. Otherwise, the cluster autoscaler might become non responsive.
- Ensure all pods have a defined resource request—to function correctly, the cluster autoscaler needs a specified resource request. This is because the cluster autoscaler makes decisions based on the status of pods and utilization of individual nodes, and can be thrown off if the calculation is off.

[1] Minikube Handbook
[2] Horizontal Pod Autoscaling
[3] Vertical Pod Autoscaling
[4] Scaling Kubernetes Clusters
[5] KEDA: Kubernetes Event-Driven Autoscaling

Thanks!