Contribute to this list by updating this Google sheet

Cluster Autoscaling

Automatically adding new worker nodes to increase overall resources (cpu, memory, disk, network).

Name	Description
Cluster Autoscaler	Scales nodes up and down in a Kubernetes cluster. Works on AWS, Azure and GCP. It scales nodes count by modifying the autoscaling group. In the background it constantly checks to see if pods on the cluster are failing to start because of insufficient resources and if they are it will trigger a scaling event. This autoscaler will wait until you have exhausted every resource and then begin to scale. It will also scale down when a certain threshold of resources are idle.
Kubernetes EC2 Autoscaler	Node-level autoscaler for Kubernetes on AWS EC2 that is designed for batch jobs. The key metric used for scaling is the number of pending scheduled pods. This is more predictable if you know that spinning up a certain number of pods requires a certain number of nodes due to consistent resource usage on each. It also has a couple of nice features like AWS multi region support and the ability to safely drain nodes when scaling down.
Kube AWS Autoscaler	Based on some frustration with the Cluster Autoscaler somebody decided to make a better one but specifically for AWS. One frustration is that the Cluster Autoscaler only scales when it's too late i.e. resources are fully exhausted. It also doesn't respect AZ placement so you could end up with an unbalanced cluster in terms of fault tolerance. The kube-aws-autoscaler also supports scaling multiple ASG's.
K8s AWS Autoscaler	AWS Kubernetes nodes autoscaler.
Jupyterhub Autoscaler	JupyterHub Notebooks are used frequently by data scientists working across large amounts of data. This autoscaler will scale a cluster based on workload defined in a notebook.
Cerebral	Kubernetes cluster autoscaler with pluggable metrics backends and scaling engines, as an alternative to the cluster-autoscaler project.
Escalator	Escalator is a batch or job optimized horizontal autoscaler for Kubernetes. Ensures jobs have completed before terminating nodes and aims to scale clusters as quickly as possible to meet batch demands. AWS only currently.

Pod Autoscaling

Horizontally increasing the number of pods or vertically increasing the resources a pod is given.

Name	Description
HPA	The Horizontal Pod Autoscaler (HPA) automatically scales the number of pods in a replication controller, deployment or replica set based on observed CPU utilization (or, with custom metrics support, on some other application-provided metrics).
VPA	The Vertical Pod Autoscaler (VPA) automatically adjusts the amount of CPU and memory requested by pods running in the Kubernetes Cluster. Not to be used alongside HPA in a cluster (unless HPA is scaling on custom metrics only).
Addon Resizer	A simplified version of VPA that modifies resource requests of a deployment based on the number of nodes in the Kubernetes Cluster.
K8s Prom HPA	Before HPA supported custom metrics this was a great way to scale based on Prometheus monitoring data.
Cluster Proportional Autoscaler	Pods will proportionately increase or decrease with cluster size. Whereas with the HPA you must pre-configure CPU and memory on all containers, the cluster-proportional-autoscaler doesn't need this as an input and will calculate it dynamically.
Kube AMQP Autoscaler	Scale pods based on AMQP message queue
K8s Rabbit Pod Autoscaler	Scale pods based on RabbitMQ messages
Kube SQS Autoscaler	Scale pods based on AWS SQS messages
Kube Cloudwatch Autoscaler	This is a Kubernetes deployment that will manage the autoscaling of one other Kubernetes deployment/replica/pod, periodically scaling the number of replicas based on any AWS CloudWatch metric (ex: SQS Queue Size or Max Age, ELB Response Time, etc). An example would be using it to increase the number of pods when the age of the oldest message on SQS gets too old, and decrease the number of pods when it stabilizes again.
Simple Scaler	The HPA has several shortcomings which can be listed as follows. The scaling is not restricted in terms of how pods are started and stopped at the same time. The scaling action is not based on historical data but on current usage. The thresholds for scaling up and scaling down are the same. This project aims to solve these issues.
Prometheus Kubernetes Autoscaler	Scale Kubernetes deployments based on Prometheus alerts. The autoscaler is registered as a webhook alertmanager receiver. When an alert is forwarded to this receiver the alertmanager will contact the autoscaler.
CronHPA	Cron Horizontal Pod Autoscaler(CronHPA) enables us to auto scale workloads (those that support scale subresource, e.g. deployment, statefulset) periodically using crontab scheme.
KEDA	Scales pods based on Kafka, RabbitMQ or some Azure services. More scaling backends planned.

Follow us

Tell us about a new Kubernetes application Submit

Newsletter

Never miss a thing! Sign up for our newsletter to stay updated.

Email address

Join now

About

Discover and learn about everything Kubernetes

Navigation

Privacy Policy
Contact
Submit

Follow

Twitter
LinkedIn