Autoscaling Workloads

With autoscaling, you can automatically update your workloads in one way or another. This allows your cluster to react to changes in resource demand more elastically and efficiently.

In Kubernetes, you can *scale* a workload depending on the current demand of resources. This allows your cluster to react to changes in resource demand more elastically and efficiently.

When you scale a workload, you can either increase or decrease the number of replicas managed by the workload, or adjust the resources available to the replicas in-place.

The first approach is referred to as *horizontal scaling*, while the second is referred to as *vertical scaling*.

There are manual and automatic ways to scale your workloads, depending on your use case.

Scaling workloads manually

Kubernetes supports *manual scaling* of workloads. Horizontal scaling can be done using the `kubectl` CLI. For vertical scaling, you need to *patch* the resource definition of your workload.

See below for examples of both strategies.

- **Horizontal scaling**: Running multiple instances of your app
- **Vertical scaling**: Resizing CPU and memory resources assigned to containers

Scaling workloads automatically

Kubernetes also supports *automatic scaling* of workloads, which is the focus of this page.

The concept of *Autoscaling* in Kubernetes refers to the ability to automatically update an object that manages a set of Pods (for example a Deployment).

Scaling workloads horizontally

In Kubernetes, you can automatically scale a workload horizontally using a *HorizontalPodAutoscaler* (HPA).

It is implemented as a Kubernetes API resource and a controller and periodically adjusts the number of replicas in a workload to match observed resource utilization such as CPU or memory usage.

There is a walkthrough tutorial of configuring a HorizontalPodAutoscaler for a Deployment.

Scaling workloads vertically

FEATURE STATE: `Kubernetes v1.25 [stable]`

You can automatically scale a workload vertically using a *VerticalPodAutoscaler* (VPA). Unlike the HPA, the VPA doesn't come with Kubernetes by default, but is a separate project that can be found on GitHub.

Once installed, it allows you to create CustomResourceDefinitions (CRDs) for your workloads which define *how* and *when* to scale the resources of the managed replicas.

> **Note:** You will need to have the Metrics Server installed to your cluster for the HPA to work.

At the moment, the VPA can operate in four different modes:

Mode	Description
`Auto`	Currently, `Recreate` might change to in-place updates in the future
`Recreate`	The VPA assigns resource requests on pod creation as well as updates them on existing pods by evicting them when the requested resources differ significantly from the new recommendation
`Initial`	The VPA only assigns resource requests on pod creation and never changes them later.
`Off`	The VPA does not automatically change the resource requirements of the pods. The recommendations are calculated and can be inspected in the VPA object.

Requirements for in-place resizing

FEATURE STATE: `Kubernetes v1.27 [alpha]`

Resizing a workload in-place **without** restarting the Pods or its Containers requires Kubernetes version 1.27 or later. Additionally, the `InPlaceVerticalScaling` feature gate needs to be enabled.

`InPlacePodVerticalScaling`: Enables in-place Pod vertical scaling.

Autoscaling based on cluster size

For workloads that need to be scaled based on the size of the cluster (for example `cluster-dns` or other system components), you can use the *Cluster Proportional Autoscaler*. Just like the VPA, it is not part of the Kubernetes core, but hosted as its own project on GitHub.

The Cluster Proportional Autoscaler watches the number of schedulable nodes and cores and scales the number of replicas of the target workload accordingly.

If the number of replicas should stay the same, you can scale your workloads vertically according to the cluster size using the *Cluster Proportional Vertical Autoscaler*. The project is **currently in beta** and can be found on GitHub.

While the Cluster Proportional Autoscaler scales the number of replicas of a workload, the Cluster Proportional Vertical Autoscaler adjusts the resource requests for a workload (for example a Deployment or DaemonSet) based on the number of nodes and/or cores in the cluster.

Event driven Autoscaling

It is also possible to scale workloads based on events, for example using the *Kubernetes Event Driven Autoscaler* (**KEDA**).

KEDA is a CNCF graduated enabling you to scale your workloads based on the number of events to be processed, for example the amount of messages in a queue. There exists a wide range of adapters for different event sources to choose from.

Autoscaling based on schedules

Another strategy for scaling your workloads is to **schedule** the scaling operations, for example in order to reduce resource consumption during off-peak hours.

Similar to event driven autoscaling, such behavior can be achieved using KEDA in conjunction with its `Cron` scaler. The `Cron` scaler allows you to define schedules (and time zones) for scaling your workloads in or out.

Scaling cluster infrastructure

If scaling workloads isn't enough to meet your needs, you can also scale your cluster infrastructure itself.

Scaling the cluster infrastructure normally means adding or removing nodes. Read cluster autoscaling for more information.

What's next

- Learn more about scaling horizontally

    - Scale a StatefulSet
    - HorizontalPodAutoscaler Walkthrough
- Resize Container Resources In-Place
- Autoscale the DNS Service in a Cluster
- Learn about cluster autoscaling

Feedback

Was this page helpful?

Yes No

Last modified February 18, 2024 at 2:59 PM PST: Add concept page about cluster autoscaling (b39e01b971)