Autoscaling in Kubernetes

What is Autoscaling?

Autoscaling

- **Autoscaling**: increase and decrease **the capacity** of our workloads **without human intervention.**

- *Workload autoscaling*: the automated management of the capacity for individual workloads.
- *Cluster autoscaling*: the automated management of the capacity of the underlying platform that hosts workloads.

Autoscaling: increase and decrease the capacity of our workloads without human intervention.

- **Horizontal scaling(scale in/out):**
  - Scale the number of Pods for a particular application or the number of nodes in a cluster that hosts applications.
  - Suitable for applications with frequent, significant changes in load prefers horizontal scaling.

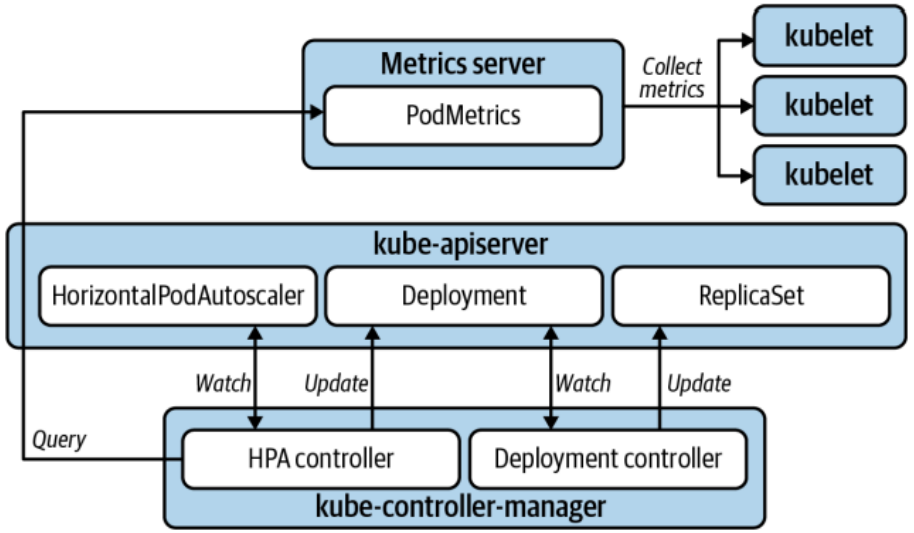
- **Vertical scaling(scale up/down) :**
  - For an application, this is changing the **resource requests and/or limits for the containers** of the application.
  - For nodes of a cluster, this generally involves changing **the amount of CPU and memory resources available**.
  - **VPA requires a restart for the application.**

	Horizontal	**Vertical**
Node	add or remove nodes	change nodes' CPU or memory
Pod	add or remove pods	change pods' CPU or memory

Workload Autoscaling

Horizontal Pod Autoscaler

- *The **Horizontal Pod Autoscaler** changes the shape of your Kubernetes workload by automatically increasing or decreasing the number of Pods in response to the workload demand(cpu or memory consumption, custom metrics,..).*
- In the common way, **HPA uses Kubernetes Metrics Server to get the PodMetrics.**
- The Metrics Server collects CPU and memory usage metrics for containers from the kubelets in the cluster and makes them available through the resource metrics API in PodMetrics resources.



1. ***Metrics server*** gets metrics from kubelet.
2. ***HPA controller*** query metrics from Metrics server every 15 seconds.
3. ***HPA controller*** calculates the number of desired replicas.
4. ***HPA controller*** updates the relevant Deployment through the API server.
5. ***Deployment controller*** responds by updating the ReplicaSet, which leads to a change in the number of Pods..

Horizontal Pod Autoscaling Example

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: php-apache
spec:
  selector:
    matchLabels:
      run: php-apache
  replicas: 1
  template:
    metadata:
      labels:
        run: php-apache
    spec:
      containers:
      - name: php-apache
        image: k8s.gcr.io/hpa-example
        ports:
        - containerPort: 80
        resources:
          limits:
            cpu: 500m
          requests:
            cpu: 200m
```

```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  targetCPUUtilizationPercentage: 50
```

```
NAME         REFERENCE                     TARGET    MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache/scale   0% / 50%  1         10        1          18s
```

```
NAME         REFERENCE                     TARGET      MINPODS   MAXPODS   REPLICAS   AGE
php-apache   Deployment/php-apache/scale   305% / 50%  1         10        7          3m

NAME         READY   UP-TO-DATE   AVAILABLE   AGE
php-apache   7/7     7            7           19m
```

HPA Limitations

Although HPA is widely applicable and relatively uncomplicated to implement, but it still has some limitations:

- *Not all workloads can scale horizontally*
  - For applications that cannot share load among distinct instances (e.g., some stateful workloads and leader-elected applications). These use cases may consider Vertical Pod autoscaling.
- *The cluster size will limit scaling*
  - It may lead to run out of capacity available in the worker nodes of a cluster. This can be solved by provisioning sufficient capacity ahead of time, using alerts to **prompt your platform operators to add capacity manually, or by using Cluster Autoscaling.**
- ***CPU and memory may not be the right metric to use for scaling decisions***
  - If your workload exposes a custom metric that better identifies a need to scale, it can be used.