Autoscale

Getting ready

- Get a small service that runs a CPU-intensive process.

  ```
  kubectl run php-apache --image=k8s.gcr.io/hpa-example --
  requests=cpu=200m --limits=cpu=500m --expose --port=80
  ```

- Set up an autoscaler

  ```
  kubectl autoscale deployment php-apache --cpu-percent=50
  --min=1 --max=10
  ```

Send some traffic (different console)

- Create another service.

  `kubectl run -it load-generator --image=busybox /bin/sh`

- Send some requests from your pod.

```
while true;
do wget -qO- http://php-apache.default.svc.cluster.local;
done
```

How autoscaler works

- Calculates the number of replicas as follows:
  `numReplicas = numReplicas*(currentKPI/desiredKPI)`
- Support for multiple metrics simultaneously, choosing the maximum number of replicas.

Other autoscalers

- Vertical autoscaler is planned, but not yet available.
- Cluster autoscale can be defined at cluster creation.
    - When nodes start creating pods on 'Pending' state, K8s spins more nodes.

Overcommitted state

- This happens if you have a node where the sum of all container limits is higher than resources available.
- For CPU, Kubernetes will give containers their request and throttle the rest.

Overcommitted state (cont.)

- For memory, there need to be decisions on which containers to terminate.
  - Containers with no requests get terminated.
  - Containers above requests, even within the limit.
  - Pods within their requests, if critical system components (`kubelet`, `docker`) start to take more resources.

References

- Documentation
- Resource limits best practices