Horizontal Pod Autoscaling with Kubernetes

Overview

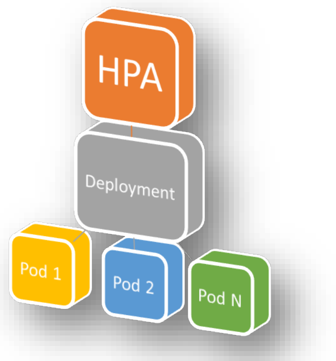
So your developers have finalized the latest version of the application, containerized it in a docker container, and are ready for production! The Operations team has decided to use Kubernetes to orchestrate and manage the resources. The site goes live and is a huge success…too big of one, and your pods maxed out and the application can't keep up! You don't want to keep an admin watching resources 24 hours a day to adjust for peaks and valleys of usage, so what do you do? You enable horizontal pod automation, and let those poor admins go home and get some sleep.

How does it work?

As the name implies, horizontal pod automation, manages the number of pods based upon a pre-defined metric. In most cases it is tied to CPU or memory usage, but it can also be tied to custom, multiple or external metrics. The HPA is a control loop, that checks the system every 15 seconds (default value, but can be changed) to make sure it is in compliance with the desired state, and if not, adjusts the pods to get as close to that as possible. At a high level, the algorithm that is followed is:

```
desiredReplicas = ceil[currentReplicas * (currentMetricValue / desiredMetricValue)]
```

For example, if our desired CPU usage was 50% (desiredMetricValue) and the current CPU usage is 100% (CurrentMetricValue), then the number of replicas will be doubled, since 100/50 = 2. As usage goes down (25%), then the replicas will be halved since 25/50 = 0.5. As new pods are started and become available (usually within seconds for linux-based containers), the HPA factors the new pods into the ratio to adjust the number of pods. As the ratio decreases, the HPA intentionally delays pod shutdown (default is five minutes) to reduce system thrashing and provide a graceful scale down.



How is it configured?

Once your Kubernetes environment is set up and running your docker container, enabling HPA is a very simple process. Once the Kubernetes cluster is set up and exposed (in our case named php-apache[1]), use the following command to create the autoscaler:

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

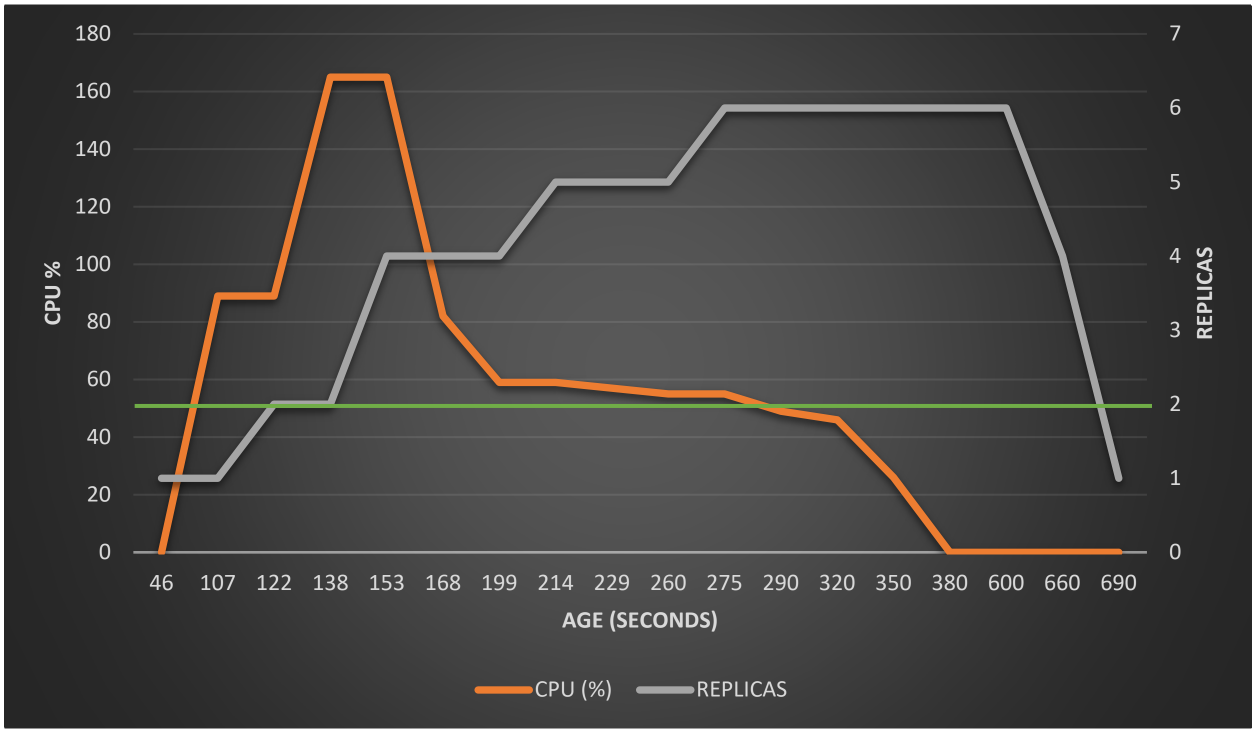
The above command references the existing cluster (php-apache), defines the target CPU usage at 50%, and sets the minimum (1) and maximum (10) number of pods. The specifications or pod limits can be adjusted to meet the expected needs of the application and server configuration. To verify that HPA has been enabled, run:

[1] Kubernetes sample php-apache docker image - https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/

```
$ kubectl get hpa
NAME          REFERENCE                TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
php-apache    Deployment/php-apache    0%/50%    1         10        1          19s
```

The "kubectl get hpa" command shows the current CPU usage (0%) over the target CPU usage (50%), the minimum and maximum number of pods specified, and the current number of replicas (pods).

We will now simulate a load on the cluster to watch how the HPA manages the number of pods to stay as close to the specified ideal state as possible. The following chart tracks CPU utilization (orange line) and replica creation/deletion (gray line) over time as the load is applied, and then removed. The green line is the desired CPU load of 50%.



Starting at 46 seconds, the load generator started causing the CPU usage to rise, quickly going past the desired 50% utilization. At 122 seconds, the HPA identified that the CPU load was approximately 90%; utilizing the above ratio (90/50 is approximately 2), the replicas were doubled to 2. As the CPU load continued to rise, the HPA continued to poll the usage every 15 seconds to increase the number of replicas. At 153 seconds, the CPU usage was at 165% per pod, causing the HPA to continue to incrementally spin up new pods until the per pod utilization dropped below 50% (at approximately 290 seconds). As the load was removed, the HPA began to slowly and gracefully shut down pods as to not create instability in the system, reverting back to one pod at 690 seconds.

The horizontal pod autoscaler allows your system to smoothly, quickly, and most importantly automatically handle varying loads. HPA could also stand up new pods when existing ones are sick or have lost communication.

About OnPoint

OnPoint Consulting, Inc. (OnPoint) delivers secure IT infrastructure, enterprise systems, cybersecurity and program management solutions for the U.S. federal government. Our specialized strategy, cyber and technology capabilities are changing the way our clients improve performance, effectively deliver results and manage risk. OnPoint holds ISO 9001:2015, ISO 20000-1:2011, ISO 27001:2013 certifications and a CMMI Maturity Level 3 rating.

OnPoint is a part of the Publicis Sapient platform, with access to industry leading AI tools and teams. Contact us at innovation@onpointcorp.com or visit onpointcorp.com to learn more about us and our services.