

How does the Horizontal Pod Autoscaler work?

The Horizontal Pod Autoscaler 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).

a . Run & expose and Application

1. `$ kubectl run <your-deployment-name> --image=asyed755/ril-load-testing:latest --requests=cpu=200m`

2. `$ kubectl expose deployment <your-service-name> --type=LoadBalancer --port=80`

b . Create Horizontal Pod Autoscaler

Now that the server is running, we will create the autoscaler using `kubectl autoscale`. The following command will create a Horizontal Pod Autoscaler that maintains between 1 and 10 replicas of the Pods.

Roughly speaking, HPA will increase and decrease the number of replicas (via the deployment) to maintain an average CPU utilization across all Pods of 50%

1. `$ kubectl autoscale deployment <your-deployment-name> --cpu-percent=20 --min=1 --max=10`

2. `$ kubectl get hpa`

NAME	REFERENCE	TARGET	MINPODS	MAXPODS	REPLICAS	AGE
ril	Deployment/ril/scale	0% / 20%	1	10	1	18s

Please note that the current CPU consumption is 0% as we are not sending any requests to the server (the CURRENT column shows the average across all the pods controlled by the corresponding deployment).

c . Increase load

Now, we will see how the autoscaler reacts to increased load. We will start a container, and send an infinite loop of queries to the **<your-service-name>** service (please run it in a different terminal):

1. `$ kubectl run -i --tty load-generator-<your-name> --image=busybox /bin/sh`

Note: If you want to login to a already deployed load generated do

`$ kubectl attach load-generator-7bbbb4fdd4-g697m -c load-generator -i -t`

`load-generator-7bbbb4fdd4-g697m` is the pod name ---

Run the below command to get the load-generator pod name

`$ kubectl get pods --all-namespaces # to get the load-generator pod name`

Hit enter **for command** prompt

2. `$ while true; do wget -q -O- <LoadBalancer EndPoint url of the application >; done`

Run the Below command to the LoadBalancer endpoint of the service

3. `$ kubectl get services ## to get the end LB point of the the application`

```
arshad@arshad-Latitude-E6330: ~$ kubectl get services
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes    ClusterIP     100.64.0.1      <none>           443/TCP          3h
rill          LoadBalancer 100.71.117.26   aaacc9ecff8c11e88c9902b53799273-1329021564.us-east-1.elb.amazonaws.com 80:31276/TCP 7m
```

Now run the below command (replace the load-balancer url)

3. `$ while true; do wget -q -O-`

`http://a8b8a9381ffa011e8a5460226b9b836a-1716557945.us-east-1.elb.amazonaws.com/ ;`

`done`

On the load generator terminal

Within a minute or so, we should see the higher CPU load by executing:

Meantime open **two more terminals** and run the below commands

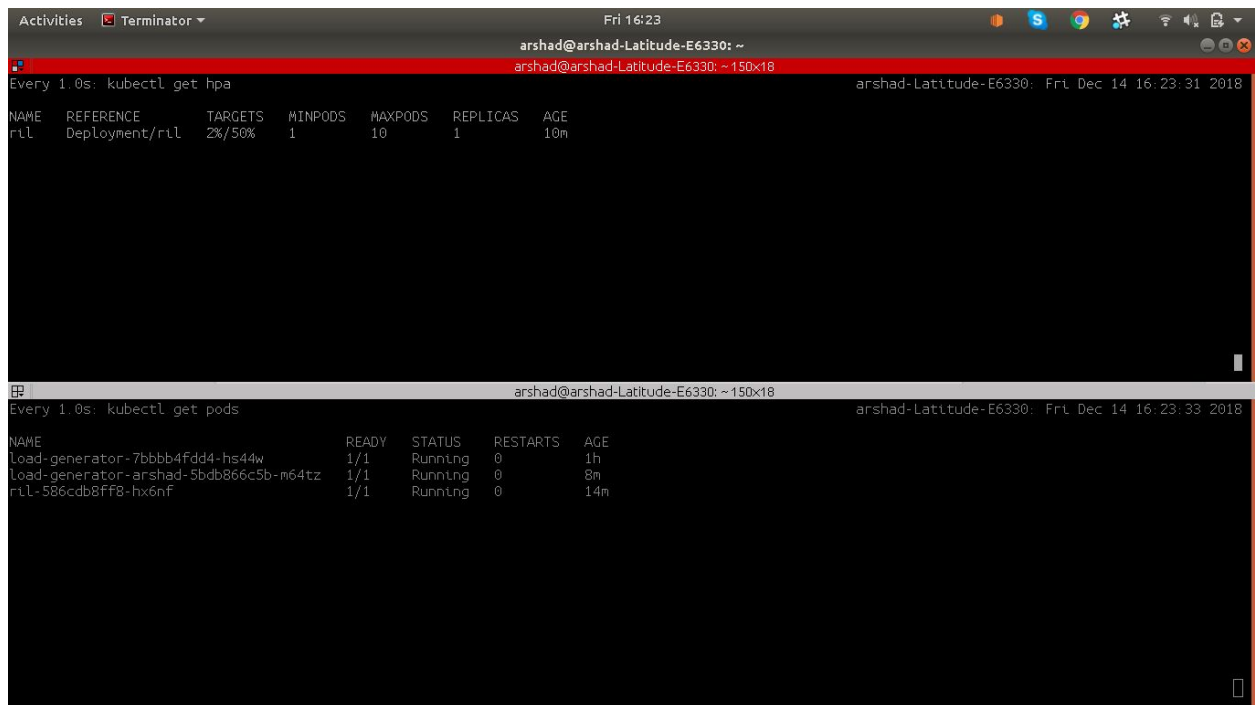
On terminal one run the below command

4. \$ watch -n 1 kubectl get hpa

On terminal two run the below command

5. \$ watch -n 1 kubectl get pods

It should look something like below



```
arshad@arshad-Latitude-E6330: ~  
Every 1.0s: kubectl get hpa  
NAME      REFERENCE      TARGETS  MINPODS  MAXPODS  REPLICAS  AGE  
ril       Deployment/ril  2%/50%   1         10        1          10m  
  
arshad@arshad-Latitude-E6330: ~  
Every 1.0s: kubectl get pods  
NAME                                READY   STATUS    RESTARTS   AGE  
load-generator-7bbbb4fdd4-hs44w     1/1     Running   0           1h  
load-generator-arshad-5bdb866c5b-m64tz 1/1     Running   0           8m  
ril-586cdb8ff8-hx6nf                1/1     Running   0           14m
```

6. \$ kubectl get hpa

NAME	REFERENCE	TARGET	CURRENT	MINPODS	MAXPODS	REPLICAS	AGE
ril	Deployment/ril/scale	305% / 20%	305%	1	10	1	3m

[illegible]

7. \$ kubectl get deployment rli

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
ril	7	7	7	7	19m

Login to the Dashboard and you will be able to observe that the application has started scaling horizontally.

Note. Once the cpu (or any other metric) goes below the defined threshold, the HPA will automatically scale down the application to minimum.