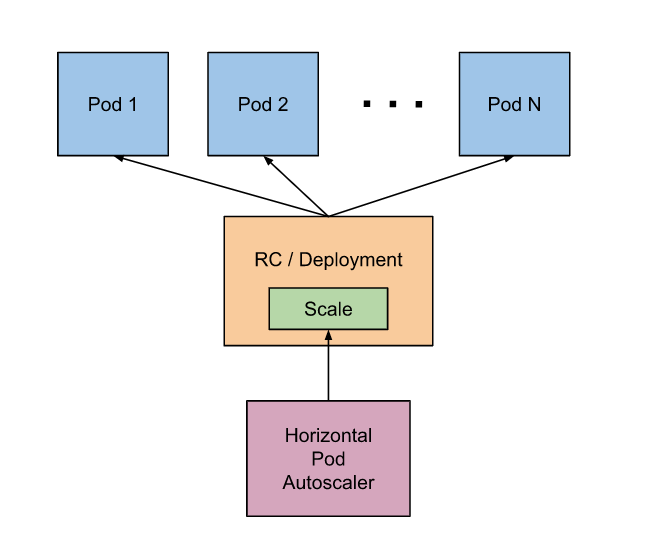
The **Horizontal Pod Autoscaler** automatically scales the number of Pods in a replication controller, deployment, replica set or stateful set based on observed CPU utilization (or, with [custom metrics](https://git.k8s.io/community/contributors/design-proposals/instrumentation/custom-metrics-api.md) support, on some other application-provided metrics).



The Horizontal Pod Autoscaler is implemented as a control loop, with a period controlled by the controller manager's --horizontal-pod-autoscaler-sync-period .

During each period, the controller manager queries the resource utilization against the metrics specified in each HorizontalPodAutoscaler definition.

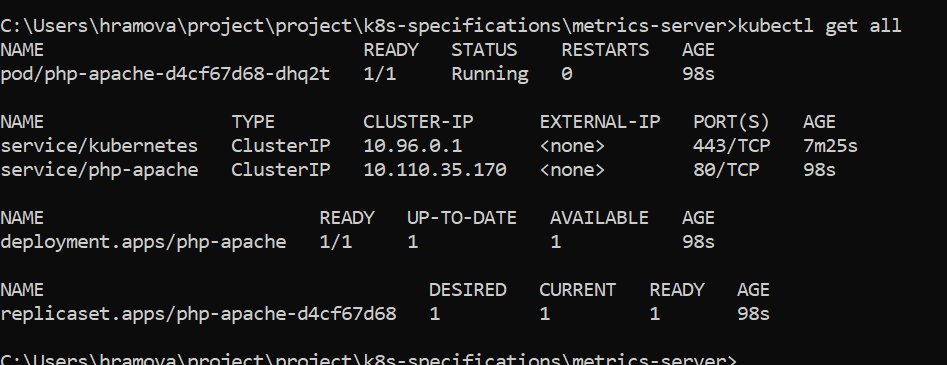
Prerequests: Deploy metrics-server

kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>

**Important enabling metrics**: minikube addons enable metrics-server

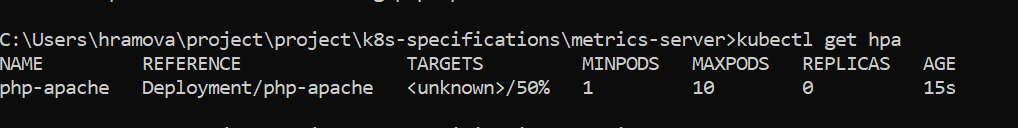
To demonstrate Horizontal Pod Autoscaler we will use a custom docker image based on the php-apache image.

kubectl apply -f https://k8s.io/examples/application/php-apache.yaml

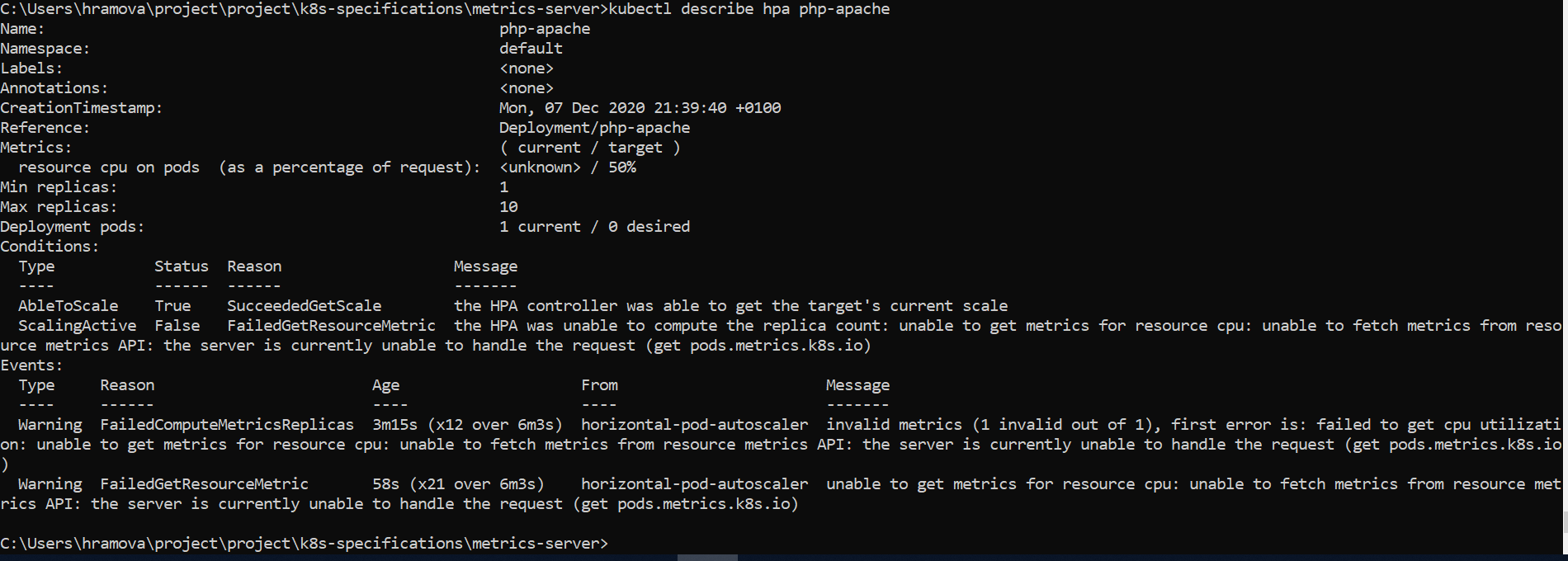


Now that the server is running, we will create the autoscaler using [kubectl autoscale](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands" \l "autoscale). The following command will create a Horizontal Pod Autoscaler that maintains between 1 and 10 replicas of the Pods controlled by the php-apache deployment. 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% (since each pod requests 200 milli-cores by kubectl run), this means average CPU usage of 100 milli-cores). See [here](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#algorithm-details) for more details on the algorithm.

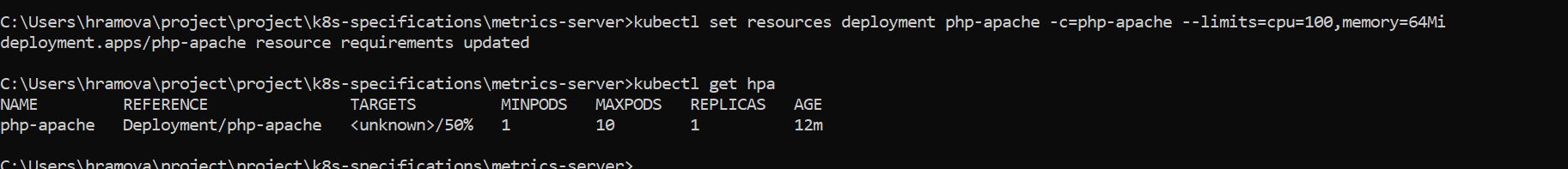
kubectl autoscale deployment php-apache --cpu-percent=50 --min=1 --max=10



Target should be 0%/50%.



Possible fix : Set limits, however it didn’t help



Another possible fix:

$ minikube addons disable metrics-server

$ kubectl create -f deploy/1.8+/

# edit metric-server deployment to add the flags

# args:

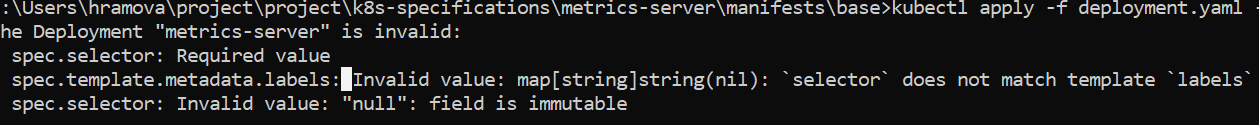
# - --kubelet-insecure-tls

# - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname

$ kubectl edit deploy -n kube-system metrics-server

<https://stackoverflow.com/questions/53725248/how-to-enable-kubeapi-server-for-hpa-autoscaling-metrics/53727101#53727101>

<https://stackoverflow.com/questions/64598763/kubernetes-hpa-unable-to-get-metrics-for-resource-memory-no-metrics-returned>



So, here somest he problem that I can’t modify the yaml file.

What was going to happen further considering if there want’s this error:

Let’s see how the autoscaler reacts to increased load. We will start a container, and send an infinite loop of queries to the php-apache service (please run it in a different terminal):

kubectl run -i --tty load-generator --rm --image=busybox --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://php-apache; done"

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

kubectl get hpa

NAME REFERENCE TARGET MINPODS MAXPODS REPLICAS AGE

php-apache Deployment/php-apache/scale 305% / 50% 1 10 1 3m

Here, CPU consumption has increased to 305% of the request. As a result, the deployment was resized to 7 replicas:

kubectl get deployment php-apache

NAME READY UP-TO-DATE AVAILABLE AGE

php-apache 7/7 7 7 19m

Stop load

We will finish our example by stopping the user load.

Then we will verify the result state (after a minute or so):

NAME REFERENCE TARGET MINPODS MAXPODS REPLICAS AGE

php-apache Deployment/php-apache/scale 0% / 50% 1 10 1 11m

kubectl get deployment php-apache

NAME READY UP-TO-DATE AVAILABLE AGE

php-apache 1/1 1 1 27m

Here CPU utilization dropped to 0, and so HPA autoscaled the number of replicas back down to 1.