Roberto Morassi del Blanco

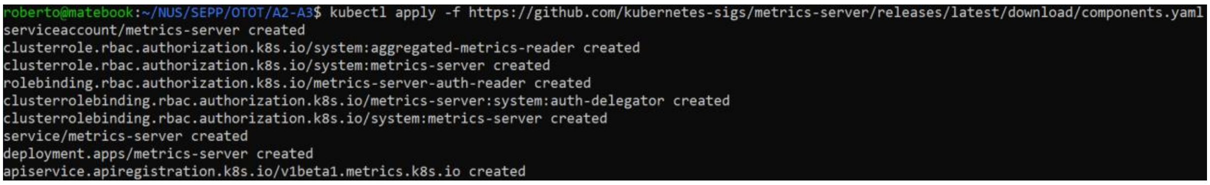
A0265239X

https://github.com/rmorassi/CS3219-OTOT-A2-A3

Task A3.1

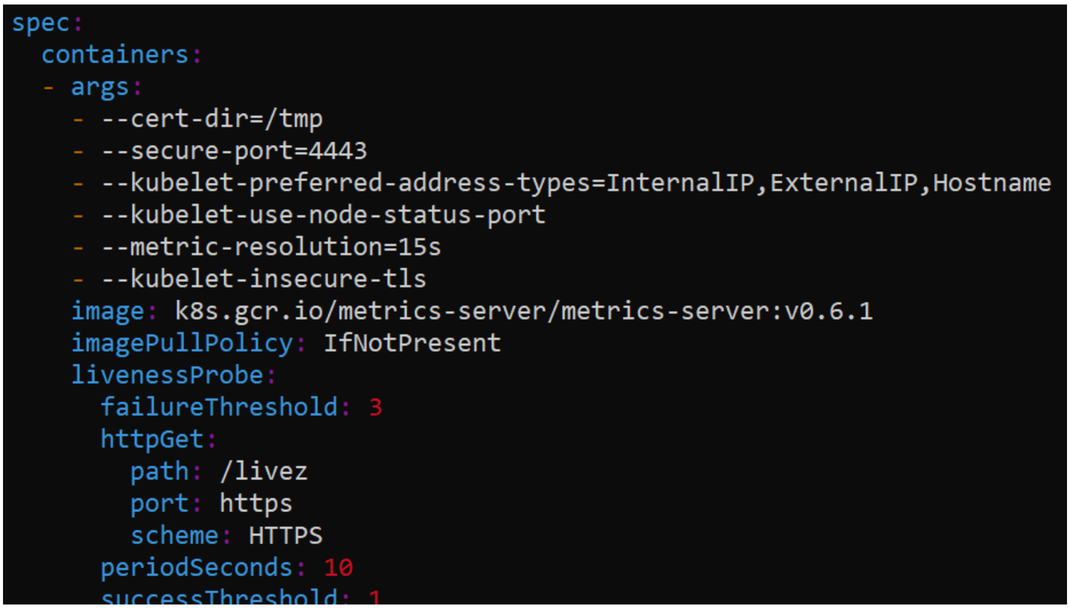
Commands to create metrics-server and verify it works

1. Run “kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml” to download the metrics-server from online and apply it to our clusters.



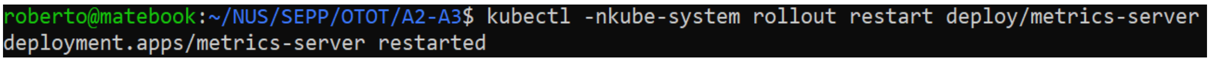
```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

2. Execute “kubectl -nkube-system edit deploy/metrics-server” to edit the metrics-server and add the “--kubelet-insecure-tls” flag to deployment.spec.containers[].args[]. Below we can see how I added it.



```
spec:
  containers:
  - args:
    - --cert-dir=/tmp
    - --secure-port=4443
    - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
    - --kubelet-use-node-status-port
    - --metric-resolution=15s
    - --kubelet-insecure-tls
    image: k8s.gcr.io/metrics-server/metrics-server:v0.6.1
    imagePullPolicy: IfNotPresent
    livenessProbe:
      failureThreshold: 3
      httpGet:
        path: /livez
        port: https
        scheme: HTTPS
      periodSeconds: 10
      successThreshold: 1
```

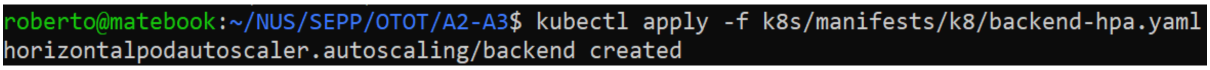
3. Run “kubectl -nkube-system rollout restart deploy/metrics-server” to restart metrics-server.



```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl -nkube-system rollout restart deploy/metrics-server
deployment.apps/metrics-server restarted
```

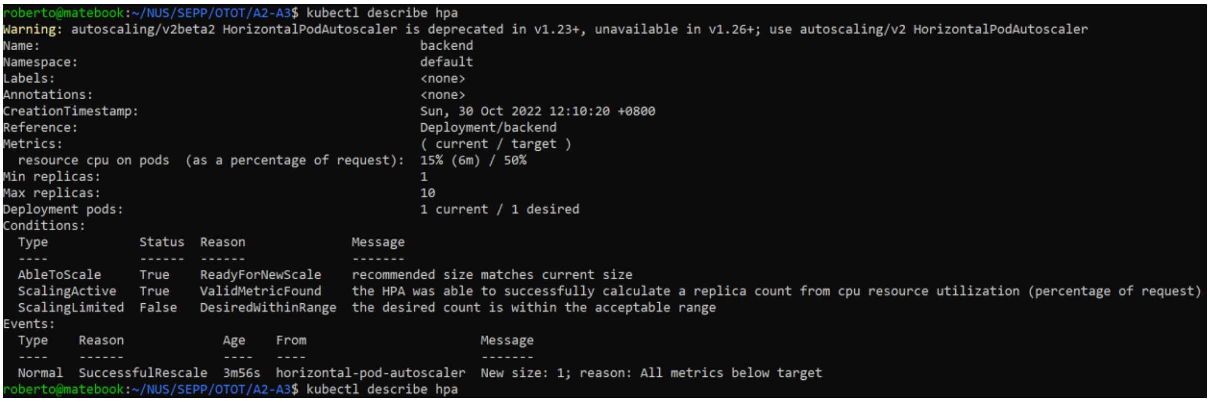
Commands to create HPA and verify it works

1. Run “kubectl apply -f k8s/manifests/k8/backend-hpa.yaml” to apply the HPA.



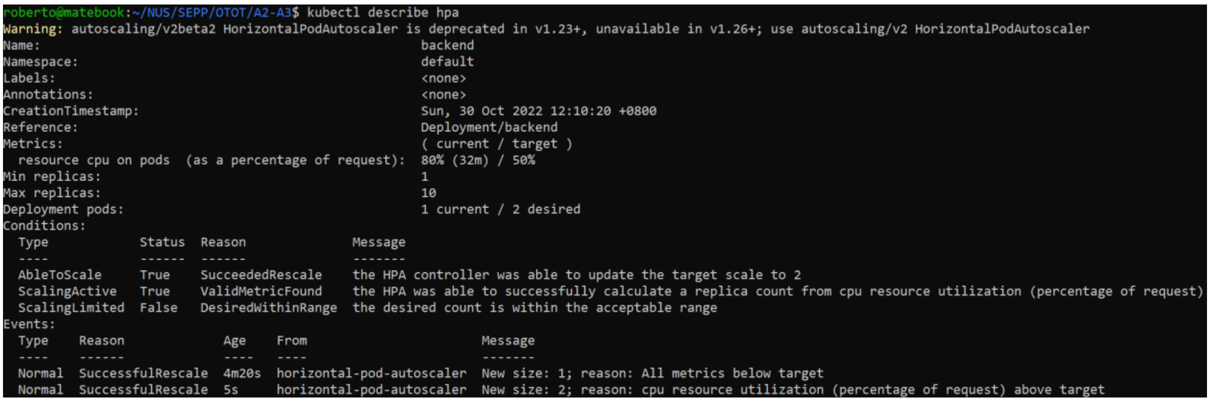
```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl apply -f k8s/manifests/k8/backend-hpa.yaml
horizontalpodautoscaler.autoscaling/backend created
```

2. To test the HPA to see if it works, I tried stressing the system in different ways
    a. I initially only queried the app intermittently to not stress the servers too much. As seen in screenshot below, as the load is below target, and the system does not need to scale up (see number of deployed pods and desired pods – stays constant).



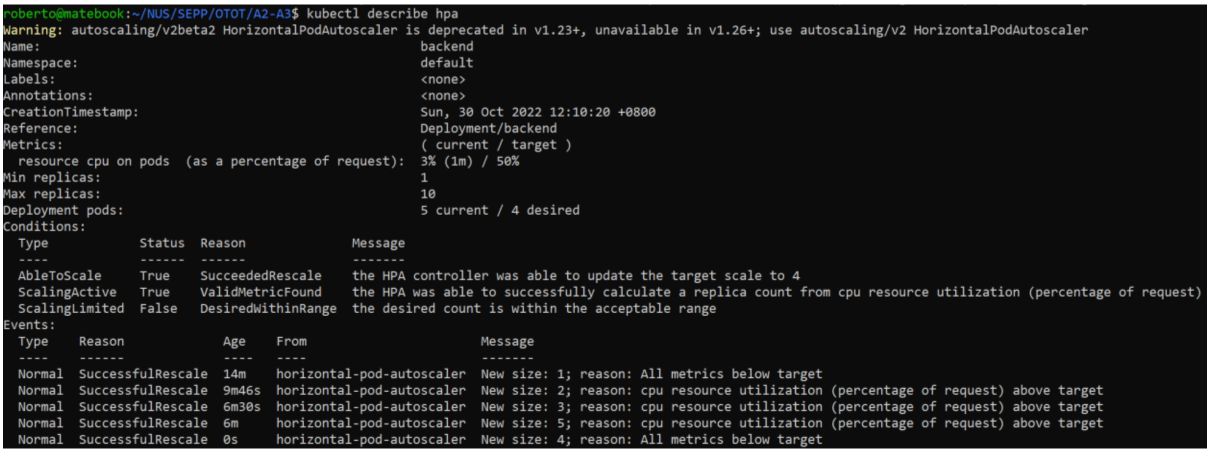
```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl describe hpa
Warning: autoscaling/v2beta2 HorizontalPodAutoscaler is deprecated in v1.23+, unavailable in v1.26+; use autoscaling/v2 HorizontalPodAutoscaler
Name:                                                  backend
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Sun, 30 Oct 2022 12:10:20 +0800
Reference:                                             Deployment/backend
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  15% (6m) / 50%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       1 current / 1 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ReadyForNewScale    recommended size matches current size
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  3m56s  horizontal-pod-autoscaler  New size: 1; reason: All metrics below target
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl describe hpa
```

    b. I then queried very the app frequently to stress the servers above the target. As seen in the screenshot below, as the load increased above the target, the system scaled up (see number of deployed pods and desired pods – wants more pods).



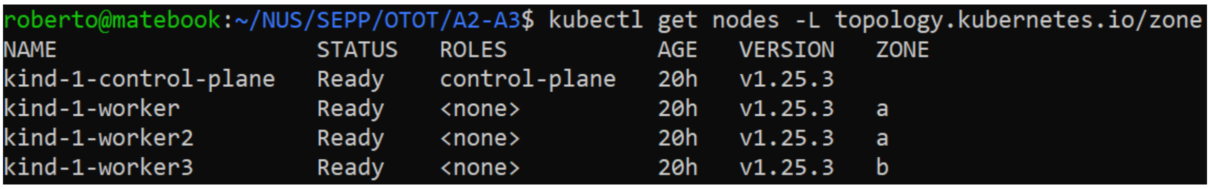
```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl describe hpa
Warning: autoscaling/v2beta2 HorizontalPodAutoscaler is deprecated in v1.23+, unavailable in v1.26+; use autoscaling/v2 HorizontalPodAutoscaler
Name:                                                  backend
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Sun, 30 Oct 2022 12:10:20 +0800
Reference:                                             Deployment/backend
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  80% (32m) / 50%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       1 current / 2 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    SucceededRescale    the HPA controller was able to update the target scale to 2
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  4m20s  horizontal-pod-autoscaler  New size: 1; reason: All metrics below target
  Normal  SuccessfulRescale  5s     horizontal-pod-autoscaler  New size: 2; reason: cpu resource utilization (percentage of request) above target
```

    c. I then stopped querying the app frequently to destress the servers below the target. As seen in the screenshot below, as the load is decreased below the target, the system scaled down (see number of deployed pods and desired pods – wants less).



```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl describe hpa
Warning: autoscaling/v2beta2 HorizontalPodAutoscaler is deprecated in v1.23+, unavailable in v1.26+; use autoscaling/v2 HorizontalPodAutoscaler
Name:                                                  backend
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Sun, 30 Oct 2022 12:10:20 +0800
Reference:                                             Deployment/backend
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  3% (1m) / 50%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       5 current / 4 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    SucceededRescale    the HPA controller was able to update the target scale to 4
  ScalingActive   True    ValidMetricFound    the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange  the desired count is within the acceptable range
Events:
  Type    Reason             Age    From                       Message
  ----    ------             ----   ----                       -------
  Normal  SuccessfulRescale  14m    horizontal-pod-autoscaler  New size: 1; reason: All metrics below target
  Normal  SuccessfulRescale  9m46s  horizontal-pod-autoscaler  New size: 2; reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  6m30s  horizontal-pod-autoscaler  New size: 3; reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  6m     horizontal-pod-autoscaler  New size: 5; reason: cpu resource utilization (percentage of request) above target
  Normal  SuccessfulRescale  0s     horizontal-pod-autoscaler  New size: 4; reason: All metrics below target
```

3. We can confirm that the pods are running in different zones by executing “kubectl get nodes -L topology.kubernetes.io/zone”

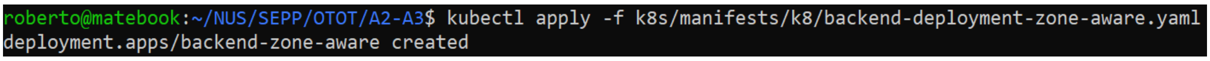


```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl get nodes -L topology.kubernetes.io/zone
NAME                    STATUS   ROLES           AGE   VERSION   ZONE
kind-1-control-plane    Ready    control-plane   20h   v1.25.3
kind-1-worker           Ready    <none>          20h   v1.25.3   a
kind-1-worker2          Ready    <none>          20h   v1.25.3   a
kind-1-worker3          Ready    <none>          20h   v1.25.3   b
```

Task A3.2

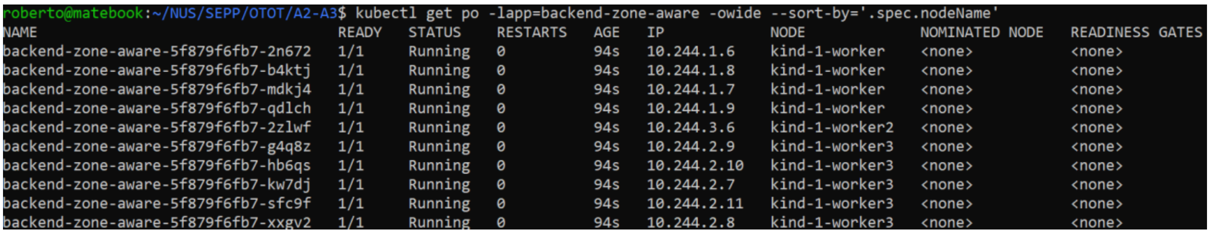
Commands to create the Deployment and verify it works

1. We apply the zone aware deployment with the command “kubectl apply -f k8s/manifests/k8/backend-deployment-zone-aware.yaml”



```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl apply -f k8s/manifests/k8/backend-deployment-zone-aware.yaml
deployment.apps/backend-zone-aware created
```

2. We recall that “worker” and “worker2” are running in zone a while “worker3” is running in zone b. We run “kubectl get po -lapp=backend-zone-aware -owide --sort-by='.spec.nodeName'” to check how many pods are assigned to each node. We can see that there are 4 pods assigned to worker, 1 to worker 2, and 5 to worker3. This means that there are 5 pods running in zone a and 5 pods in zone b, and thus we know that the pods are uniformly distributed across the two zones.



```
roberto@matebook:~/NUS/SEPP/OTOT/A2-A3$ kubectl get po -lapp=backend-zone-aware -owide --sort-by='.spec.nodeName'
NAME                                 READY   STATUS    RESTARTS   AGE   IP            NODE             NOMINATED NODE   READINESS GATES
backend-zone-aware-5f879f6fb7-2n672   1/1     Running   0          94s   10.244.1.6    kind-1-worker    <none>           <none>
backend-zone-aware-5f879f6fb7-b4ktj   1/1     Running   0          94s   10.244.1.8    kind-1-worker    <none>           <none>
backend-zone-aware-5f879f6fb7-mdkj4   1/1     Running   0          94s   10.244.1.7    kind-1-worker    <none>           <none>
backend-zone-aware-5f879f6fb7-qdlch   1/1     Running   0          94s   10.244.1.9    kind-1-worker    <none>           <none>
backend-zone-aware-5f879f6fb7-2zlwf   1/1     Running   0          94s   10.244.3.6    kind-1-worker2   <none>           <none>
backend-zone-aware-5f879f6fb7-g4q8z   1/1     Running   0          94s   10.244.2.9    kind-1-worker3   <none>           <none>
backend-zone-aware-5f879f6fb7-hb6qs   1/1     Running   0          94s   10.244.2.10   kind-1-worker3   <none>           <none>
backend-zone-aware-5f879f6fb7-kw7dj   1/1     Running   0          94s   10.244.2.7    kind-1-worker3   <none>           <none>
backend-zone-aware-5f879f6fb7-sfc9f   1/1     Running   0          94s   10.244.2.11   kind-1-worker3   <none>           <none>
backend-zone-aware-5f879f6fb7-xxgv2   1/1     Running   0          94s   10.244.2.8    kind-1-worker3   <none>           <none>
```