cloud native & ai
training &
consulting

Kubernetes

Observability Challenge Solutions

2.1. CHALLENGE: kubectl top

Using the `kubectl top` command and its help text available from `-h` or `--help`, try the following operations:

- Retrieve the current CPU and Memory utilization of all control plane components in the `kube-system` namespace by their label

Discover the control plane labels by getting the pods in the `kube-system` namespace:

```
~$ kubectl -n kube-system get po --show-labels

NAME                                         READY   STATUS    RESTARTS
AGE     LABELS
coredns-76f75df574-9k4tx                     1/1     Running   0
101m    k8s-app=kube-dns,pod-template-hash=76f75df574
coredns-76f75df574-9p29r                     1/1     Running   0
101m    k8s-app=kube-dns,pod-template-hash=76f75df574
etcd-ip-172-31-4-161                         1/1     Running   0
101m    component=etcd,tier=control-plane
kube-apiserver-ip-172-31-4-161               1/1     Running   0
101m    component=kube-apiserver,tier=control-plane
kube-controller-manager-ip-172-31-4-161      1/1     Running   0
101m    component=kube-controller-manager,tier=control-plane
kube-proxy-kbdll                             1/1     Running   0
101m    controller-revision-hash=5f6677ccc4,k8s-app=kube-proxy,pod-
template-generation=1
kube-scheduler-ip-172-31-4-161               1/1     Running   0
101m    component=kube-scheduler,tier=control-plane
metrics-server-c5fbf4cb9-lr2kd               1/1     Running   0
3m10s   k8s-app=metrics-server,pod-template-hash=c5fbf4cb9
weave-net-x76wg                              2/2     Running   1 (98m ago)
98m     controller-revision-hash=d94c4d6bc,name=weave-net,pod-template-
generation=1

~$
```

The control plane components have a common label: `tier=control-plane`; use it to filter pods

```
~$ kubectl -n kube-system top pods -l tier=control-plane

NAME                                        CPU(cores)   MEMORY(bytes)
etcd-ip-172-31-4-161                        21m          42Mi
kube-apiserver-ip-172-31-4-161              47m          281Mi
kube-controller-manager-ip-172-31-4-161     15m          50Mi
kube-scheduler-ip-172-31-4-161              4m           20Mi

~$
```

- Sort the output by memory use

```
~$ kubectl -n kube-system top pods -l tier=control-plane --sort-by memory

NAME                                        CPU(cores)   MEMORY(bytes)
kube-apiserver-ip-172-31-4-161              51m          281Mi
kube-controller-manager-ip-172-31-4-161     17m          50Mi
etcd-ip-172-31-4-161                        22m          42Mi
kube-scheduler-ip-172-31-4-161              4m           20Mi

~$
```

- Which control plane component is currently using the most memory?

Based on the example, the `kube-apiserver` pod. It depends on which one is sorted to the top

- Retrieve a listing of resource use of all nodes in your cluster sorted by CPU utilization without printing the column headers

```
~$ kubectl top nodes --no-headers --sort-by cpu

ip-172-31-4-161   201m   10%   1782Mi   47%

~$
```

3.1. CHALLENGE: HPA editing

Before you clean up, try the following operations:

- Adjust the deployment or HPA so that the `web1` deployment scales when a pod reaches 50m of CPU:

Adjusting the deployment and leaving the HPA alone: Change the cpu request in the web1 deployment to `100m`

```
~/hpa$ nano target-deploy.yaml && cat target-deploy.yaml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: web1
  name: web1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: web1
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: web1
    spec:
      containers:
      - image: rxmllc/target
        imagePullPolicy: Never
        name: target
        resources:
          requests:
            cpu: "100m"
```

```
~/hpa$ kubectl apply -f target-deploy.yaml

deployment.apps/web1 configured

~/hpa$
```

OR:

Adjusting the HPA and leaving the deployment alone: Edit the HPA to scale at 25% CPU:

```
~/hpa$ kubectl edit hpa web1
```

```yaml
...

spec:
  maxReplicas: 5
  metrics:
  - resource:
      name: cpu
      target:
        averageUtilization: 25
        type: Utilization
    type: Resource
status:
  conditions:
```

```
:wq

horizontalpodautoscaler.autoscaling/web1 edited

~/hpa$ kubectl get hpa

NAME   REFERENCE         TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
web1   Deployment/web1   0%/25%    1         5         1          100m

~/hpa$
```

- Ensure the HPA can scale the target deployment up to 3 replicas

Edit the HPA so its maxReplicas is 3:

```
~/hpa$ kubectl edit hpa web1
```

```
spec:
  maxReplicas: 3
  metrics:
  - resource:
      name: cpu
      target:
        averageUtilization: 25
        type: Utilization
    type: Resource
  minReplicas: 1
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: web1
```

```
:wq

horizontalpodautoscaler.autoscaling/web1 edited

~/hpa$ kubectl get hpa
```

```
NAME   REFERENCE         TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
web1   Deployment/web1   0%/25%    1         3         1          23m

~/hpa$
```

- Adjust the HPA minimum count to 2 replicas:

```
~/hpa$ kubectl get pods

NAME                     READY   STATUS    RESTARTS      AGE
driver                   1/1     Running   1 (12m ago)   15m
web1-658f9667d6-w4c82    1/1     Running   0             24m

~/hpa$ kubectl get deploy web1

NAME   READY   UP-TO-DATE   AVAILABLE   AGE
web1   1/1     1            1           24m

~/hpa$ kubectl edit hpa web1
```

```yaml
...

spec:
  maxReplicas: 3
  metrics:
  - resource:
      name: cpu
      target:
        averageUtilization: 25
        type: Utilization
    type: Resource
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: web1
```

```
:wq

horizontalpodautoscaler.autoscaling/web1 edited

~/hpa$ kubectl get deploy web1 -w

NAME   READY   UP-TO-DATE   AVAILABLE   AGE
web1   1/1     1            1           24m
web1   1/2     1            1           25m
```

```
web1   1/2     1            1           25m
web1   1/2     1            1           25m
web1   1/2     2            1           25m
web1   2/2     2            2           25m

^C

~/hpa$ kubectl get pods

NAME                    READY   STATUS    RESTARTS      AGE
driver                  1/1     Running   1 (13m ago)   17m
web1-658f9667d6-hjnph   1/1     Running   0             20s
web1-658f9667d6-w4c82   1/1     Running   0             25m

~/hpa$
```

- What happens to the deployment?

The deployment's replica count is adjusted up to match the HPA, effectively enforcing one-way scaling.