.

Lesson 03 Demo 09

Creating and Configuring the Metrics Server

Objective: To create and configure the metrics server in the Kubernetes cluster to identify the top nodes, pods, and containers

Tools required: kubeadm, kubectl, kubelet, and containerd

Prerequisites: A Kubernetes cluster (refer to Demo 01 from Lesson 01 for setting up a cluster)

Steps to be followed:

1. Create a deployment
2. Configure the metrics server
3. Verify the metrics server deployment

Step 1: Create a deployment

1.1 On the master node, enter the following command to create a YAML file that will define the deployment:
nano deployment.yaml

```
labsuser@master:~$ nano deployment.yaml
labsuser@master:~$
```

1.2 Enter the following code in the **deployment.yaml** file:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
```

.

```
    tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

```
  GNU nano 6.2                                                    deployment.yaml *
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/gb-frontend:v3
```

1.3 Execute the following command to create a deployment:
kubectl create -f deployment.yaml

```
labsuser@master:~$ kubectl create -f deployment.yaml
deployment.apps/frontend created
labsuser@master:~$
```

.

1.4 Execute the following commands to get the status of the deployment:

kubectl get deployment frontend

kubectl get rs

kubectl get pods -l tier=frontend

```
labsuser@master:~$ kubectl get deployment frontend
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
frontend   3/3     3            3           5m51s
labsuser@master:~$ kubectl get rs
NAME                   DESIRED   CURRENT   READY   AGE
frontend-5b85744f5d    3         3         3       6m44s
labsuser@master:~$ kubectl get pods -l tier=frontend
NAME                          READY   STATUS    RESTARTS   AGE
frontend-5b85744f5d-bjkwz     1/1     Running   0          7m3s
frontend-5b85744f5d-q55bd     1/1     Running   0          7m3s
frontend-5b85744f5d-sbvlr     1/1     Running   0          7m3s
labsuser@master:~$
```

1.5 Describe the deployment using the following command:

kubectl describe deploy/frontend

```
labsuser@master:~$ kubectl describe deploy/frontend
Name:                   frontend
Namespace:              default
CreationTimestamp:      Mon, 06 Nov 2023 18:41:20 +0000
Labels:                 app=guestbook
                        tier=frontend
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               tier=frontend
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  tier=frontend
  Containers:
   php-redis:
    Image:        gcr.io/google_samples/gb-frontend:v3
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   frontend-5b85744f5d (3/3 replicas created)
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  11m   deployment-controller  Scaled up replica set frontend-5b85744f5d to 3
labsuser@master:~$
```

.

Step 2: Configure the metrics server

2.1 Run the following command to create the metrics server:
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

```
labsuser@master:~$ 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
labsuser@master:~$
```

2.2 Verify the status of the metrics server using the following command:
kubectl get pods -n kube-system

```
labsuser@master:~$ kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-7ddc4f45bc-hf2pb   1/1     Running   0          33m
calico-node-kb7rr                          1/1     Running   0          32m
calico-node-n5cwx                          1/1     Running   0          32m
calico-node-tvhps                          1/1     Running   0          33m
coredns-5dd5756b68-9gksj                   1/1     Running   0          34m
coredns-5dd5756b68-p55fr                   1/1     Running   0          34m
etcd-master.example.com                    1/1     Running   0          34m
kube-apiserver-master.example.com          1/1     Running   0          34m
kube-controller-manager-master.example.com 1/1     Running   0          34m
kube-proxy-4qgn9                           1/1     Running   0          32m
kube-proxy-6n2lj                           1/1     Running   0          32m
kube-proxy-nsblp                           1/1     Running   0          34m
kube-scheduler-master.example.com          1/1     Running   0          34m
metrics-server-fbb469ccc-5bhhl             0/1     Running   0          93s
labsuser@master:~$
```

Note: The metrics server is not in a ready state.

2.3 Run the following command to fetch the **k8s-metrics-server.patch.yaml** file:
wget -c https://gist.githubusercontent.com/initcron/1a2bd25353e1faa22a0ad41ad1c01b62/raw/008e23f9fbf4d7e2cf79df1dd008de2f1db62a10/k8s-metrics-server.patch.yaml

```
labsuser@master:~$ wget -c https://gist.githubusercontent.com/initcron/1a2bd25353e1faa22a0ad41ad1c01b62/raw/008e23f9fbf4d7e2cf79df1dd008de2f1db62a10/k8s-metrics-server.patch.yaml
--2023-11-06 19:07:33--  https://gist.githubusercontent.com/initcron/1a2bd25353e1faa22a0ad41ad1c01b62/raw/008e23f9fbf4d7e2cf79df1dd008de2f1db62a10/k8s-metrics-server.patch.yaml
Resolving gist.githubusercontent.com (gist.githubusercontent.com)... 185.199.111.133, 185.199.108.133, 185.199.109.133, ...
Connecting to gist.githubusercontent.com (gist.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 205 [text/plain]
Saving to: ‘k8s-metrics-server.patch.yaml’

k8s-metrics-server.patch.yaml                 100%[===================================================================================================>]     205  --.-KB/s    in 0s

2023-11-06 19:07:33 (8.00 MB/s) - ‘k8s-metrics-server.patch.yaml’ saved [205/205]

labsuser@master:~$
```

.

2.4 Run the following command to view the content of the **k8s-metrics-server.patch.yaml** file:
cat k8s-metrics-server.patch.yaml

```
labsuser@master:~$ cat k8s-metrics-server.patch.yaml
spec:
  template:
    spec:
      containers:
      - name: metrics-server
        command:
        - /metrics-server
        - --kubelet-insecure-tls
        - --kubelet-preferred-address-types=InternalIPlabsuser@master:~$
```

2.5 Run the following command to deploy the metrics server:
kubectl patch deploy metrics-server -p "$(cat k8s-metrics-server.patch.yaml)" -n kube-system

```
labsuser@master:~$ kubectl patch deploy metrics-server -p "$(cat k8s-metrics-server.patch.yaml)" -n kube-system
deployment.apps/metrics-server patched
labsuser@master:~$
```

Step 3: Verify the metrics server deployment

3.1 Execute the following command to verify the status of the metrics server:
kubectl get pods -n kube-system

```
labsuser@master:~$ kubectl get pods -n kube-system
NAME                                       READY   STATUS    RESTARTS   AGE
calico-kube-controllers-7ddc4f45bc-hf2pb   1/1     Running   0          51m
calico-node-kb7rr                          1/1     Running   0          50m
calico-node-n5cwx                          1/1     Running   0          50m
calico-node-tvhps                          1/1     Running   0          51m
coredns-5dd5756b68-9gksj                   1/1     Running   0          52m
coredns-5dd5756b68-p55fr                   1/1     Running   0          52m
etcd-master.example.com                    1/1     Running   0          52m
kube-apiserver-master.example.com          1/1     Running   0          52m
kube-controller-manager-master.example.com 1/1     Running   0          52m
kube-proxy-4qgn9                           1/1     Running   0          50m
kube-proxy-6n2lj                           1/1     Running   0          50m
kube-proxy-nsblp                           1/1     Running   0          52m
kube-scheduler-master.example.com          1/1     Running   0          52m
metrics-server-678d4b775-7lmwz             1/1     Running   0          5m26s
labsuser@master:~$
```

The metrics server is now running.

.

3.2 Execute the following commands to sort all nodes and identify those with top memory and CPU usage in the cluster:

kubectl top nodes
kubectl top nodes --sort-by cpu
kubectl top nodes --sort-by memory
kubectl top nodes master.example.com

```
labsuser@master:~$ kubectl top nodes
NAME                        CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master.example.com          673m         33%    2534Mi          67%
worker-node-1.example.com   225m         11%    2166Mi          28%
worker-node-2.example.com   232m         11%    2193Mi          28%
labsuser@master:~$ kubectl top nodes --sort-by cpu
NAME                        CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master.example.com          462m         23%    2538Mi          67%
worker-node-2.example.com   234m         11%    2209Mi          28%
worker-node-1.example.com   181m         9%     2172Mi          28%
labsuser@master:~$ kubectl top nodes --sort-by memory
NAME                        CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master.example.com          453m         22%    2538Mi          67%
worker-node-2.example.com   223m         11%    2194Mi          28%
worker-node-1.example.com   195m         9%     2175Mi          28%
labsuser@master:~$ kubectl top nodes master.example.com
NAME                CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master.example.com  474m         23%    2540Mi          67%
labsuser@master:~$
```

By following these steps, you have successfully configured the metric server in the Kubernetes cluster to identify the top nodes, pods, and containers.