27Oct2022
Day**16**
ETCD
Horizontal Pod Autoscaling-HPA
---

etcd, K8s Database

https://etcd.io/
what is **ETCD**?
-A distributed, reliable key-value store for the most critical data of a distributed system.
-etcd is a strongly consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines

on **MasterNode**
hostname
master1.example.com
hostname -i
192.168.29.104
ls /etc/kubernetes/manifests/
etcd.yaml kube-apiserver.yaml kube-controller-manager.yaml kube-scheduler.yaml
vim /etc/kubernetes/manifests/etcd.yaml
etcd port number ->2379, 2380
etcd database location ->/var/lib/etcd
ls /var/lib/etcd/member/snap/
db
etcd pub/private key + ca-certification
ls /etc/kubernetes/pki/etcd/
ca.crt ca.key healthcheck-client.crt healthcheck-client.key peer.crt peer.key server.crt server.key

Interact with ETCD

https://github.com/etcd-io/etcd/releases
mkdir etcd
cd etcd
wget https://github.com/etcd-io/etcd/releases/download/v3.5.5/etcd-v3.5.5-linux-amd64.tar.gz
pwd
/root/etcd
ls
etcd-v3.5.5-linux-amd64.tar.gz
tar xfv etcd-v3.5.5-linux-amd64.tar.gz
cp -r etcd-v3.5.5-linux-amd64/etcd* /usr/local/bin/
ls /usr/local/bin/
etcdctl --help
etcd --version
etcd Version: 3.5.5
Git SHA: 19002cfc6
Go Version: go1.16.15
Go OS/Arch: linux/amd64
etcdctl version
etcdctl version: 3.5.5
API version: 3.5

Backup ETCD

step1-make activity on K8s cluster
kubectl create deployment dpl1 --image nginx --replicas 3
kubectl create deployment dpl2 --image redis --replicas 3
kubectl create namespace space1
kubectl get deployments.apps
kubectl get ns
kubectl get pods
step2-take backup
to take backup need pub-key, private-key and certification but, here we use K8s cluster etcd own pub-key, private-key and certification.
etcdctl --help
etcdctl version
API version: **3**.5

ls /etc/kubernetes/pki/etcd/
ca.crt ca.key healthcheck-client.crt healthcheck-client.key peer.crt peer.key server.crt server.key
backup
ETCDCTL_API=3 etcdctl snapshot save --endpoints=127.0.0.1:2379 --cert="/etc/kubernetes/pki/etcd/server.crt" --key="/etc/kubernetes/pki/etcd/server.key" --cacert="/etc/kubernetes/pki/etcd/ca.crt" /opt/etcd27Oct22.db
ls /opt
etcd27Oct22.db
du -h /opt/etcd27Oct22.db
5.4M /opt/etcd27Oct22.db

status

```
# etcdctl snapshot status /opt/etcd27Oct22.db
# etcdutl snapshot status /opt/etcd27Oct22.db
b73d8ba7, 7570, 1810, 5.6 MB
# etcdctl snapshot status /opt/etcd27Oct22.db --write-out="table"
# etcdutl snapshot status /opt/etcd27Oct22.db --write-out="table"
+----------+----------+------------+------------+
|HASH   |REVISION | TOTAL KEYS | TOTAL SIZE |
+----------+----------+------------+------------+
| b73d8ba7 |     7570 |      1810 |     5.6 MB |
+----------+----------+------------+------------+
```

Restore ETCD

step1-delete some resources

```
# kubectl get deployments.apps
# kubectl get ns
# kubectl delete deployments.apps dpl1 dpl2 --force --grace-period=0
# kubectl delete namespaces space1 --force --grace-period=0
```

restore

current ->/var/lib/etcd/member/snap/db

backup ->/opt/etcd27Oct22.db

verify

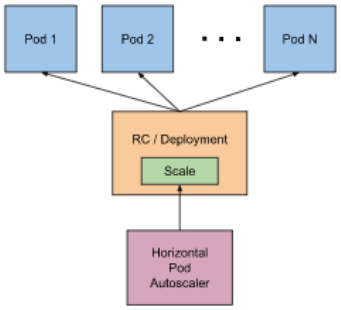
```
# cat /etc/kubernetes/manifests/etcd.yaml
- hostPath:
    path: /var/lib/etcd                                    ->current etcd db location connecting to K8s cluster kube-api

# ETCDCTL_API=3 etcdctl snapshot restore --endpoints=127.0.0.1:2379 --cert="/etc/kubernetes/pki/etcd/server.crt" --
key="/etc/kubernetes/pki/etcd/server.key" --cacert="/etc/kubernetes/pki/etcd/ca.crt" --data-dir=/var/lib/newetcd/ /opt/etcd27Oct22.db
# ll /var/lib
drwx------  3 root     root     4096 Oct 27 13:39 etcd/
drwx------  3 root     root     4096 Oct 27 14:45 newetcd/
# vim /etc/kubernetes/manifests/etcd.yaml
76   - hostPath:
 77      path: /var/lib/newetcd                            ->update etcd about new db location
:wq!
# kubectl get nodes
NOTE: it takes time
# kubectl get deployments.apps
# kubectl get ns
```

Horizontal Pod Autoscaling-HPA

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

-a HorizontalPodAutoscaler automatically updates a workload resource, with the aim of automatically scaling the workload to match demand.

-Horizontal scaling means that the response to increased load is to deploy more Pods.



Procedure

-config Metrics-Server on K8s cluster

-deploy an application, specify CPU and MEMORY on deployment.

Now, if Metrics-Server detects suppose 90% of CPU used then new pod will create.

Implement HPA on K8s Cluster

CPU-based

steps

-enable Metrics-Server

-deploy an application in deployment format

-create service, expose it

-enable HPA

-simulate load on deployment and verify HPA

-enable Metrics-Server

https://github.com/kubernetes-sigs/metrics-server

```
# kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
# kubectl edit deployments.apps --namespace kube-system metrics-server
      - --kubelet-preferred-address-types=InternalIP                    ->modify the line, keep InternalIP
       - --kubelet-insecure-tls                                         ->append line
:wq!
# watch kubectl get nodes
# kubectl top nodes
```

-deploy an application in deployment format

```
# kubectl create deployment dpl1 --image nginx --replicas 2 -o yaml --dry-run=client >dpl1.yaml
# vim dpl1.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: dpl1
  name: dpl1
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginxdpl
  template:
    metadata:
      labels:
        app: nginxdpl
    spec:
      containers:
      - image: nginx
        name: nginxcnt
        resources:
          requests:
            cpu: "5m"
            memory: "8Mi"
          limits:
            cpu: "10m"
            memory: "12Mi"
:wq!
# kubectl create -f dpl1.yaml --dry-run=client
# kubectl create -f dpl1.yaml
# kubectl get deployments.apps
# kubectl get pods
# kubectl top pod -A
# kubectl top pod --namespace default
```

-create service, expose it

```
# kubectl expose deployment dpl1 --name dpl1ext --port 80 --protocol TCP --type NodePort
# kubectl get svc
dpl1ext     NodePort    10.103.230.181   <none>      80:32254/TCP   11s
```

verify

open web browser

http://192.168.29.104:32254/

Welcome to nginx!

-enable HPA

```
# kubectl autoscale deployment dpl1 --max 6 --min 2 --cpu-percent 20 -o yaml --dry-run=client >dpl1hpa.yaml
# cat dpl1hpa.yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  creationTimestamp: null
  name: dpl1
spec:
  maxReplicas: 6
  minReplicas: 2
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: dpl1
  targetCPUUtilizationPercentage: 20
status:
  currentReplicas: 0
  desiredReplicas: 0
:wq!
# kubectl autoscale deployment dpl1 --max 6 --min 2 --cpu-percent 20
# kubectl get hpa
NAME   REFERENCE          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
dpl1   Deployment/dpl1   0%/20%    2         6         2          27s
# kubectl describe hpa
```

-simulate load on deployment and verify HPA

install **Siege** traffic emulator

https://www.joedog.org/siege-home/

https://github.com/JoeDog/siege

Siege is an open-source regression test and benchmark utility. It can stress test a single URL with a user defined number of simulated users, or it can read many URLs into memory and stress them simultaneously.

RHEL

-enable EPEL repository

https://docs.fedoraproject.org/en-US/epel/

-Install siege

```
# yum install siege -y
```

https://blog.eldernode.com/install-siege-on-centos/

UBUNTU

```
# apt-get install siege
# siege --help
```

-Emulate traffic through siege for nginx deployment

duplicate MasterNode terminal:

terminal1

```
# watch kubectl get pods
```

terminal2

```
# watch kubectl top pods
```

terminal3, generate terrific

```
# siege -q -c5 2m http://192.168.29.104:32254/
```

Now, check terminal1 and 2

number of pods will increase automatically

Now, stop siege with press Ctrl+c

and again, check terminal1

after around 3min number of pods will decrease to default.

Mohammad Ali Naghval
ali@prodevans.com