Local K8S deployment with Microk8s

Table of contents

1 Microk8s installation

The easiest and fastest way to create a local cluster is using microk8s. It's possible to make containers, push them, and deploy them directly in the laptop.

1.1 Snap installation

Snap is needed to install microk8s.

```
$ sudo apt update

$ sudo apt install snapd
```

1.2 Microk8s installation

The easiest and fastest way to create a local cluster is using microk8s. It's possible to make containers, push them, and deploy them directly in the laptop. One line installation:

```
$ sudo snap install microk8s --classic
```

After a few seconds, microk8s is installed. To check if kubernetes is running:

```
$ microk8s.kubectl get all --all-namespaces
```

We will see the following:

```
NAMESPACE   NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
default     service/kubernetes   ClusterIP   10.152.183.1   <none>        443/TCP   15m
```

1.3 MetalLB installation

We need a load balancer in our microk8s cluster in order to assign floating IPs to the pods.

```
$ kubectl apply -f
https://raw.githubusercontent.com/google/metallb/v0.7.3/manifests/meta
llb.yaml
```

This will deploy the required pods:

```
$ microk8s.kubectl get all --all-namespaces
```

We can see the following pods running:

```
NAMESPACE   NAME                 TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
default     service/kubernetes   ClusterIP   10.152.183.1   <none>        443/TCP   8m16s

NAMESPACE        NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
metallb-system   daemonset.apps/speaker    1         1         0       1            0           <none>          5s

NAMESPACE        NAME                         READY   UP-TO-DATE   AVAILABLE   AGE
metallb-system   deployment.apps/controller   0/1     1            0           5s

NAMESPACE        NAME                                   DESIRED   CURRENT   READY   AGE
metallb-system   replicaset.apps/controller-7cc9c87cfb   1         1         0       5s
```

1.4 Deploy Metrics Server

Metrics Server is a cluster-wide aggregator of resource usage data. First, we need to clone the Metrics Server GitHub repository:

```
$ git clone https://github.com/kubernetes-incubator/metrics-server.git
```

We can deploy the file with the following command:

```
$ microk8s.kubectl create -f deploy/1.8+/
```

After this, we can check that Metrics Server is running in the Microk8s cluster in our laptop:

```
$ microk8s.kubectl get all --all-namespaces
```

```
NAMESPACE        NAME                                      READY   STATUS              RESTARTS   AGE
kube-system      pod/metrics-server-fc6d4999b-jmxn9        0/1     ContainerCreating   0          6s
metallb-system   pod/controller-7cc9c87cfb-svlkn           1/1     Running             0          58m
metallb-system   pod/speaker-dz86t                         1/1     Running             0          58m

NAMESPACE     NAME                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)   AGE
default       service/kubernetes       ClusterIP   10.152.183.1     <none>        443/TCP   67m
kube-system   service/metrics-server   ClusterIP   10.152.183.184   <none>        443/TCP   6s

NAMESPACE        NAME                      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
metallb-system   daemonset.apps/speaker    1         1         1       1            1           <none>          58m

NAMESPACE        NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
kube-system      deployment.apps/metrics-server       0/1     1            0           6s
metallb-system   deployment.apps/controller           1/1     1            1           58m

NAMESPACE        NAME                                            DESIRED   CURRENT   READY   AGE
kube-system      replicaset.apps/metrics-server-fc6d4999b        1         1         0       6s
metallb-system   replicaset.apps/controller-7cc9c87cfb           1         1         1       58m
```

2 Configuration of the cluster

After the installation we need to configure some things before deploying any service.

2.1 MetalLB configuration

MetalLB needs the range of available IPs to assign floating IPs to the pods. The available range is showed in the interface configuration:

```
$ ifconfig
```

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::4133:8ce2:2afc:d49d  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:fe:29:ff  txqueuelen 1000  (Ethernet)
        RX packets 253413  bytes 307088860 (307.0 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 59963  bytes 3916698 (3.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

We need to create a config file for MetalLB, for example, *metallb-conf.yaml*. In this file we tell MetalLB to use the range of our machine and a Layer2 protocol.

```
apiVersion: v1
kind: ConfigMap
metadata:
  namespace: metallb-system
  name: config
data:
  config: |
    address-pools:
    - name: my-ip-space
      protocol: layer2
      addresses:
      - 10.0.2.1-10.0.2.21
```

After creating the file, we need to apply the configuration to the load balancer:

```
$ microk8s.kubectl apply -f metallb-conf.yaml
```

2.2 Metrics server configuration

The configuration of the metrics server is very easy, we just only need to add some lines (the ones inside the red box) in the metrics server deployment file. This file is in *deploy/1.8+/metrics-server-deployment.yaml*.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: metrics-server
  namespace: kube-system
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: metrics-server
  namespace: kube-system
  labels:
    k8s-app: metrics-server
spec:
  selector:
    matchLabels:
      k8s-app: metrics-server
  template:
    metadata:
      name: metrics-server
      labels:
        k8s-app: metrics-server
    spec:
      serviceAccountName: metrics-server
      volumes:
      # mount in tmp so we can safely use from-scratch images and/or
read-only containers
      - name: tmp-dir
        emptyDir: {}
      containers:
      - name: metrics-server
        image: k8s.gcr.io/metrics-server-amd64:v0.3.1
        imagePullPolicy: Always
        volumeMounts:
        - name: tmp-dir
          mountPath: /tmp
        command:
        - /metrics-server
        - --kubelet-insecure-tls
        - --kubelet-preferred-address-types=InternalIP
```

After saving the file, we need to apply the configuration:

```
$ microk8s.kubectl apply -f deploy/1.8+/metrics-server-deployment.yaml
```

At this point we should have a complete functional kubernetes cluster running inside our laptop.

3 Microk8s usage

The commands in Microk8s are the same as in kubectl but we need to add the *microk8s.* prefix to all of them. Here are some examples:

See the pods deployed on the cluster:

```
$ microk8s.kubectl get po
```

Check the services of the cluster:

```
$ microk8s.kubectl get svc
```

It is possible to deploy in microk8s an existing network service from a different cluster. We can copy and paste the deployments and services *yaml* files in a folder in our laptop. For example, we have a folder called *media-pilot-deployments/*, it is very to deploy the whole network service with this command:

```
$ microk8s.kubectl apply -f media-pilot-deployments/
```

This command will create the deployments and services, after a few seconds we will have all the pods running in Microk8s.