Microservice orchestration platforms using Kubernetes

Minikube basics

Konrad Dębiński

Introduction

At the beginning of the laboratory we were asked to gain basic knowledge by finishing simple guide https://kubernetes.io/docs/tutorials/hello-minikube/ . Step by step solution is presented below

Solution

First of all minikube had to be started with usage of *minikube start* command

```
kd@kd-VirtualBox: ~/Desktop
kd@kd-VirtualBox:~/Desktop$ minikube start
😄  minikube v1.14.0 on Ubuntu 20.04 (vbox/amd64)
✨  Using the docker driver based on existing profile
👍  Starting control plane node minikube in cluster minikube
🔄  Restarting existing docker container for "minikube" ...
❗  This container is having trouble accessing https://k8s.gcr.io
💡  To pull new external images, you may need to configure a proxy: https://mini
kube.sigs.k8s.io/docs/reference/networking/proxy/
🐳  Preparing Kubernetes v1.19.2 on Docker 19.03.8 ...
🔎  Verifying Kubernetes components...
🌟  Enabled addons: storage-provisioner, default-storageclass, dashboard
💡  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" by default
kd@kd-VirtualBox:~/Desktop$
```

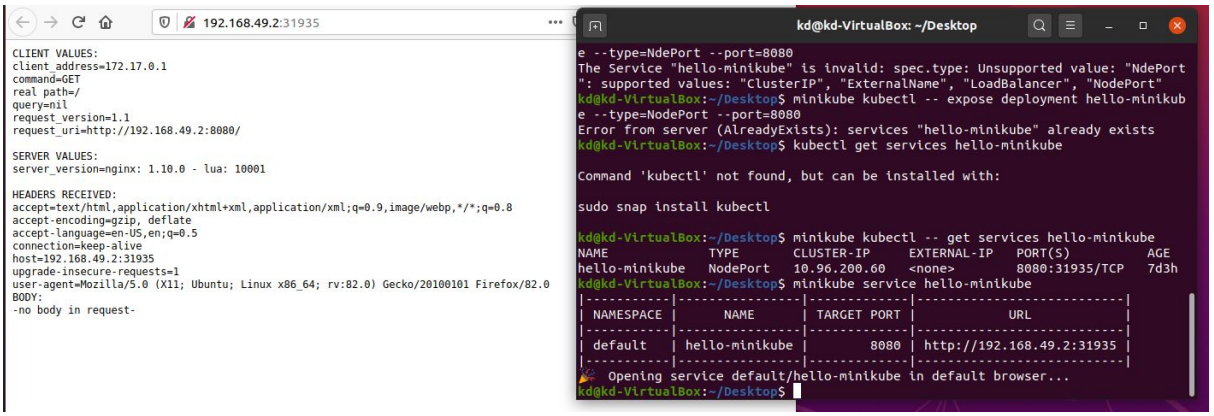
We can see that initially there are some services running as shown below

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- get po -A
NAMESPACE              NAME                                         READY   STATUS         RESTARTS   AGE
default                hello-minikube-6ddfcc9757-n4h84              1/1     Running        4          7d3h
kube-system            coredns-f9fd979d6-6qf4t                      1/1     Running        4          7d3h
kube-system            etcd-minikube                                1/1     Running        4          7d3h
kube-system            kube-apiserver-minikube                      1/1     Running        4          7d3h
kube-system            kube-controller-manager-minikube             1/1     Running        4          7d3h
kube-system            kube-proxy-6dfnc                             1/1     Running        4          7d3h
kube-system            kube-scheduler-minikube                      1/1     Running        4          7d3h
kube-system            storage-provisioner                          1/1     Running        8          7d3h
kubernetes-dashboard   dashboard-metrics-scraper-c95fcf479-5dj25    0/1     NodeAffinity   0          7d3h
kubernetes-dashboard   dashboard-metrics-scraper-c95fcf479-pwd57    0/1     NodeAffinity   0          6d23h
kubernetes-dashboard   dashboard-metrics-scraper-c95fcf479-w8nqv    1/1     Running        1          21h
kubernetes-dashboard   kubernetes-dashboard-584f46694c-7x462        0/1     NodeAffinity   0          7d3h
kubernetes-dashboard   kubernetes-dashboard-584f46694c-p2hxh        0/1     NodeAffinity   0          6d23h
kubernetes-dashboard   kubernetes-dashboard-584f46694c-xrcs8        1/1     Running        2          21h
```

With use of *kubectl get services <service_name>* we can get information about our service

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- get services hello-minikube
NAME             TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
hello-minikube   NodePort   10.96.200.60   <none>        8080:31935/TCP   7d3h
```

To start the service *minikube service hello-minikube* command has to be used. It will open up the browser windows that serves the app and shows the app's response.



What is more - the minikube tool includes a set of built-in addons that can be enabled, disabled and opened in the local Kubernetes environment.

```
kd@kd-VirtualBox:~/Desktop$ minikube addons list
|-----------------------------|----------|--------------|
|         ADDON NAME          | PROFILE  |    STATUS    |
|-----------------------------|----------|--------------|
| ambassador                  | minikube | disabled     |
| csi-hostpath-driver         | minikube | disabled     |
| dashboard                   | minikube | enabled ✅   |
| default-storageclass        | minikube | enabled ✅   |
| efk                         | minikube | disabled     |
| freshpod                    | minikube | disabled     |
| gcp-auth                    | minikube | disabled     |
| gvisor                      | minikube | disabled     |
| helm-tiller                 | minikube | disabled     |
| ingress                     | minikube | disabled     |
| ingress-dns                 | minikube | disabled     |
| istio                       | minikube | disabled     |
| istio-provisioner           | minikube | disabled     |
| kubevirt                    | minikube | disabled     |
| logviewer                   | minikube | disabled     |
| metallb                     | minikube | disabled     |
| metrics-server              | minikube | disabled     |
| nvidia-driver-installer     | minikube | disabled     |
| nvidia-gpu-device-plugin    | minikube | disabled     |
| olm                         | minikube | disabled     |
| pod-security-policy         | minikube | disabled     |
| registry                    | minikube | disabled     |
| registry-aliases            | minikube | disabled     |
| registry-creds              | minikube | disabled     |
| storage-provisioner         | minikube | enabled ✅   |
| storage-provisioner-gluster | minikube | disabled     |
| volumesnapshots             | minikube | disabled     |
|-----------------------------|----------|--------------|
```

As for exercises purposes I enabled metrics addon with use of *minikube addons enable metrics-server* command.

```
kd@kd-VirtualBox:~/Desktop$ minikube addons enable metrics-server
🌟  The 'metrics-server' addon is enabled
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- get pod,svc -n kube-system
NAME                                       READY   STATUS              RESTARTS   AGE
pod/coredns-f9fd979d6-6qf4t                1/1     Running             4          7d3h
pod/etcd-minikube                          1/1     Running             4          7d3h
pod/kube-apiserver-minikube                1/1     Running             4          7d3h
pod/kube-controller-manager-minikube       1/1     Running             4          7d3h
pod/kube-proxy-6dfnc                       1/1     Running             4          7d3h
pod/kube-scheduler-minikube                1/1     Running             4          7d3h
pod/metrics-server-d9b576748-lbhbr         0/1     ContainerCreating   0          22s
pod/storage-provisioner                    1/1     Running             8          7d3h

NAME                      TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
 AGE
service/kube-dns          ClusterIP   10.96.0.10      <none>        53/UDP,53/TCP,9153/TCP
 7d3h
service/metrics-server    ClusterIP   10.106.215.20   <none>        443/TCP
 22s
```

At the end of the guide the clean up was done. Used commands are shown below.

```
kd@kd-VirtualBox:~/Desktop$ minikube addons disable metrics-server
    "The 'metrics-server' addon is disabled
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- delete service hello-node
service "hello-node" deleted
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- delete deployment hello-node
deployment.apps "hello-node" deleted
kd@kd-VirtualBox:~/Desktop$
```

Next I had to go through another guide
https://hub.docker.com/r/strm/helloworld-http/?fbclid=IwAR03dKDS6BB-seky3b5Zh9qExP2OsCPp_2Ojt_yxCRFLvzfgU7JkB4HGu74
I had to download image which is a simple 'Hello world' in an HTTP server to be used to test load balancers. When receive an request (GET /) this image will return the current machine hostname.

It shows Hello from <hostname> for every request, making it easier to determine what host received the request.

Following the guide I created yml file and used docker-compose up command, results are presented below:

```
kd@kd-VirtualBox:~/Desktop$ nano docker-compose.yml
kd@kd-VirtualBox:~/Desktop$ docker-compose up
Creating network "desktop_default" with the default driver
Pulling front (strm/nginx-balancer:)...
latest: Pulling from strm/nginx-balancer
85b1f47fba49: Already exists
242b87a7b3c5: Pull complete
4996178133b7: Pull complete
ba1befdb4075: Pull complete
Digest: sha256:c2e775c9d7727a504fd093e21701dca04967cde169bd362a03fafc2ba3c7f226
Status: Downloaded newer image for strm/nginx-balancer:latest
Creating load-balancer  ... done
Creating desktop_web1_1 ... done
Creating desktop_web2_1 ... done
Attaching to desktop_web2_1, desktop_web1_1, load-balancer
```

Last part of this laboratory was devoted to setting up the Ingress on Minikube wit the NGINX Ingress controller. Following guide was used to finish this task: https://kubernetes.io/docs/tasks/access-application-cluster/ingress-minikube/?fbclid=IwAR3AKJTJhO0zKiljaK43jQg-_yPtIzYHJZcUIpTtks7DybhaSFA87923NIs

First of all ingress had to be enabled in addons as shown in figure below

```
kd@kd-VirtualBox:~/Desktop$ minikube addons enable ingress
🔎  Verifying ingress addon...
🌟  The 'ingress' addon is enabled
```

I checked that it is running with following command:

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- get pods -n kube-system
NAME                                        READY   STATUS      RESTARTS   AGE
coredns-f9fd979d6-6qf4t                     1/1     Running     4          7d4h
etcd-minikube                               1/1     Running     4          7d4h
ingress-nginx-admission-create-j4rwl        0/1     Completed   0          3m28s
ingress-nginx-admission-patch-cfmph         0/1     Completed   3          3m28s
ingress-nginx-controller-799c9469f7-wmr5j   1/1     Running     0          3m28s
kube-apiserver-minikube                     1/1     Running     4          7d4h
kube-controller-manager-minikube            1/1     Running     4          7d4h
kube-proxy-6dfnc                            1/1     Running     4          7d4h
kube-scheduler-minikube                     1/1     Running     4          7d4h
storage-provisioner                         1/1     Running     8          7d4h
```

Then I deployed a hello world app as shown below:

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- create deployment web --image=gcr.io/google
-samples/hello-app:1.0
deployment.apps/web created
```

Then I exposed it:

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- expose deployment web --type=NodePort --por
t=8080
service/web exposed
```

Then I verified if the service was created and is available on a node port

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- get service web
NAME    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
web     NodePort    10.110.22.165   <none>        8080:31645/TCP   25s
```

Then I visited the page:

At that point I had access to the sample app via Minikube IP address and NodePort. The next step was done to access the app using the Ingress resource.
I started by creation of yaml file with following source:

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - host: hello-world.info
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: web
                port:
                  number: 8080
```

then I created Ingress resource:

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- apply -f https://k8s.io/examples/service/ne
tworking/example-ingress.yaml
ingress.networking.k8s.io/example-ingress created
kd@kd-VirtualBox:~/Desktop$
```

After that I verified that ingress controller is directing traffic:

```
kd@kd-VirtualBox:/etc$ curl hello-world.info
Hello, world!
Version: 1.0.0
Hostname: web-79d88c97d6-4c9jm
```

After that I created and exposed second deployment:

```
kd@kd-VirtualBox:/etc$ minikube kubectl --  create deployment web2 --image=gcr.i
o/google-samples/hello-app:2.0
deployment.apps/web2 created
kd@kd-VirtualBox:/etc$ minikube kubectl -- expose deployment web2 --port=8080 --
type=NodePort
service/web2 exposed
```

and then modify yaml file in following way:

```yaml
- path: /v2
  pathType: Prefix
  backend:
    service:
      name: web2
      port:
        number: 8080
```

and applied the changes:

```
kd@kd-VirtualBox:~/Desktop$ minikube kubectl -- apply -f example-ingress.yaml
ingress.networking.k8s.io/example-ingress configured
```

Now both versions were accessible:

```
kd@kd-VirtualBox:~/Desktop$ curl hello-world.info
Hello, world!
Version: 1.0.0
Hostname: web-79d88c97d6-4c9jm
kd@kd-VirtualBox:~/Desktop$ curl hello-world.info/v2
Hello, world!
Version: 2.0.0
Hostname: web2-5d47994f45-gdx29
```

Conclusions

During this laboratory I gained a basic knowledge about Minikube deployment and configuration.