Paweł Sapek

MSK - Working with minikube

Installing Minikube

We had to use the installation process presented on this site: https://minikube.sigs.k8s.io/docs/start/

1. `curl -LO` https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
2. `sudo install minikube-linux-amd64 /usr/local/bin/minikube`
3. To start it we use

   ```
   minikube start
   ```

   This is the result

```
kubernetes@kubernetes-VirtualBox:~$ minikube start
😄  minikube v1.13.1 on Ubuntu 20.04 (vbox/amd64)
✨  Using the docker driver based on existing profile
🎉  minikube 1.14.1 is available! Download it: https://github.com/kubernetes/min
ikube/releases/tag/v1.14.1
💡  To disable this notice, run: 'minikube config set WantUpdateNotification fal
se'

👍  Starting control plane node minikube in cluster minikube
🔄  Restarting existing docker container for "minikube" ...
🐳  Preparing Kubernetes v1.19.2 on Docker 19.03.8 ...
🔎  Verifying Kubernetes components...
🌟  Enabled addons: dashboard, default-storageclass, storage-provisioner
💡  kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄  Done! kubectl is now configured to use "minikube" by default
```

To interact with it we can use the command

```
kubectl get po -A
```

which displays installed services.

```
kubernetes@kubernetes-VirtualBox:~$ kubectl get po -A
NAMESPACE              NAME                                         READY   STATUS      RESTARTS   AGE
default                web-79d88c97d6-52nk9                         1/1     Running     0          87m
default                web2-5d47994f45-nxkrs                        1/1     Running     0          69m
kube-system            coredns-f9fd979d6-8vcqb                      1/1     Running     1          14d
kube-system            etcd-minikube                                1/1     Running     0          138m
kube-system            ingress-nginx-admission-create-jd548         0/1     Completed   0          117m
kube-system            ingress-nginx-admission-patch-jss57          0/1     Completed   0          117m
kube-system            ingress-nginx-controller-789d9c4dc-bwrqs     1/1     Running     0          117m
kube-system            kube-apiserver-minikube                      1/1     Running     0          138m
kube-system            kube-controller-manager-minikube             1/1     Running     1          14d
kube-system            kube-proxy-gclnl                             1/1     Running     1          14d
kube-system            kube-scheduler-minikube                      1/1     Running     1          14d
kube-system            storage-provisioner                          1/1     Running     3          14d
kubernetes-dashboard   dashboard-metrics-scraper-c95fcf479-pvw5t    1/1     Running     1          14d
kubernetes-dashboard   kubernetes-dashboard-5c448bc4bf-rqnzm        1/1     Running     2          14d
```

Installing and testing strm/helloworld-http

We had to follow instructions from the site: https://hub.docker.com/r/strm/helloworld-http/

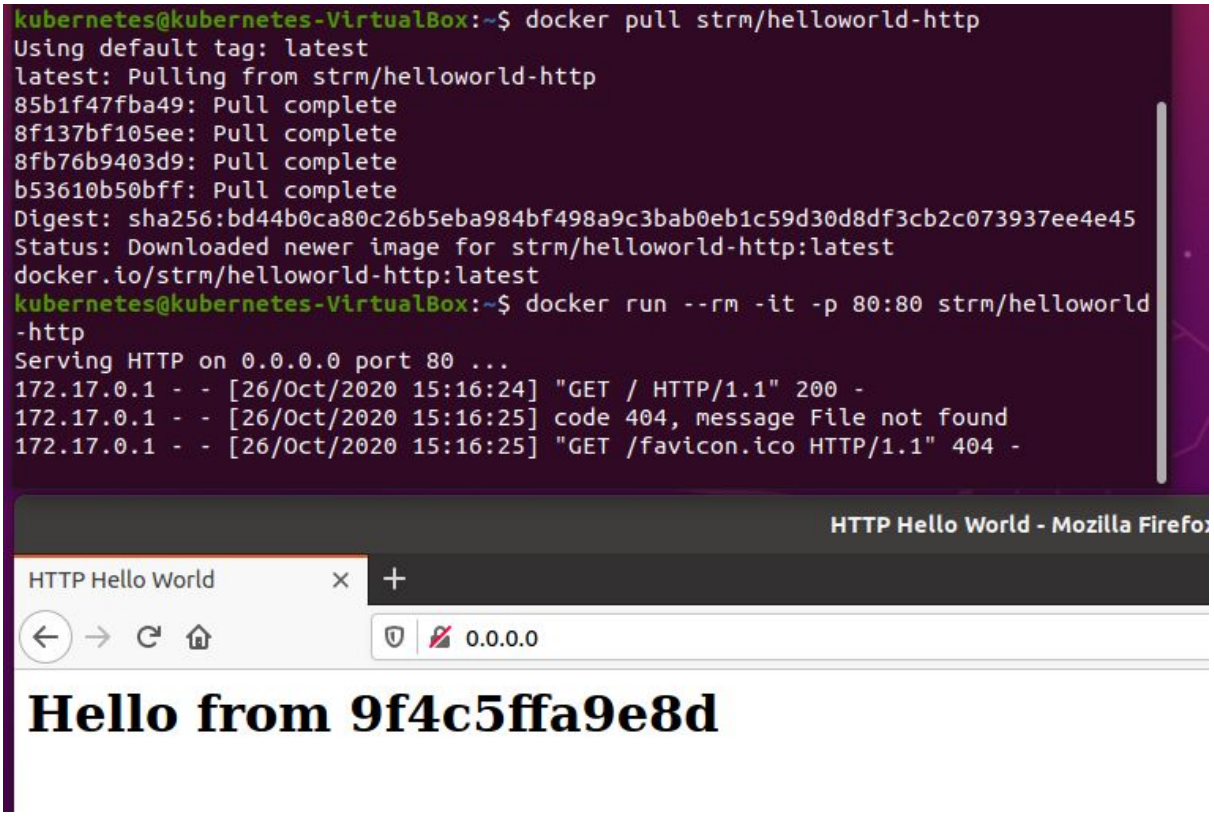
1. The first thing we had to do was to install it using the command

   `docker pull strm/helloworld-http`

2. Then we were required to run a simple test with the use of command:

   `docker run --rm -it -p 80:80 strm/helloworld-http`

   This is the result



3. Then we had to configure the load balancer. To do that we had to create a file `docker-compose.yml`
   with contained:

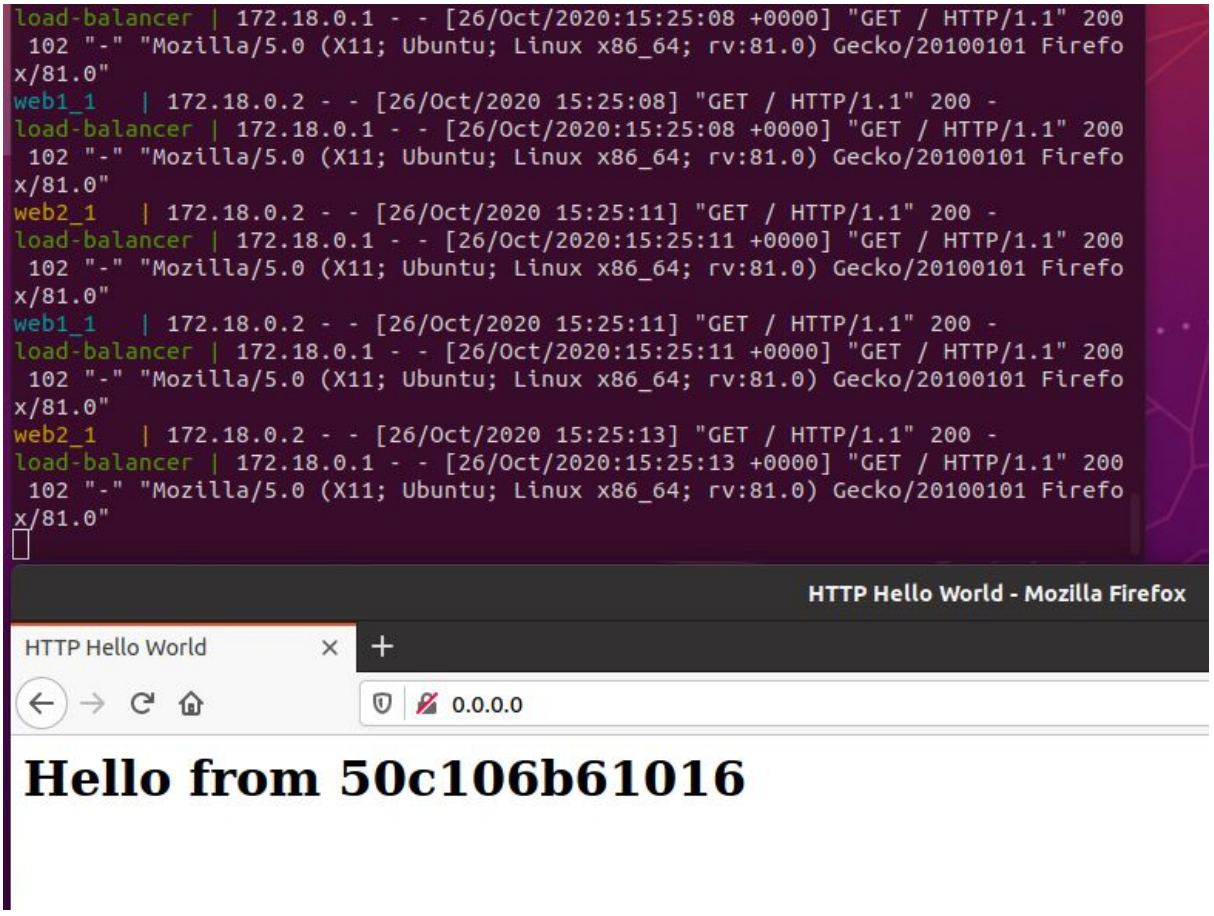
```
version: '2'
services:
    front:
        image: strm/nginx-balancer
        container_name: load-balancer
        ports:
            - "80:8080"
```

```
        environment:
            - "NODES=web1:80 web2:80"
    web1:
        image: strm/helloworld-http
    web2:
        image: strm/helloworld-http
```

4. After that we used this command to run it

```
docker-compose up
```

And the result looked like this. We can see that both load balancers are used by sending responses to GET requests, which were sent by a refreshing the website.



Set up Ingress on Minikube with the NGINX Ingress Controller

Create a Minikube cluster

1. `minikube start`

Enable the ingress controller

1. `minikube addons enable ingress`

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

2. `kubectl get pods -n kube-system`

```
kubernetes@kubernetes-VirtualBox:~$ kubectl get pods -n kube-system
NAME                                        READY   STATUS      RESTARTS   AGE
coredns-f9fd979d6-8vcqb                     1/1     Running     1          14d
etcd-minikube                               1/1     Running     0          48m
ingress-nginx-admission-create-jd548        0/1     Completed   0          27m
ingress-nginx-admission-patch-jss57         0/1     Completed   0          27m
ingress-nginx-controller-789d9c4dc-bwrqs    1/1     Running     0          27m
kube-apiserver-minikube                     1/1     Running     0          48m
kube-controller-manager-minikube            1/1     Running     1          14d
kube-proxy-gclnl                            1/1     Running     1          14d
kube-scheduler-minikube                     1/1     Running     1          14d
storage-provisioner                         1/1     Running     3          14d
```

3. `kubectl create deployment web --image=gcr.io/google-samples/hello-app:1.0`

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

4. `kubectl expose deployment web --type=NodePort --port=8080`

```
kubernetes@kubernetes-VirtualBox:~$ kubectl expose deployment web --type=NodePor
t --port=8080
service/web exposed
```

5. `kubectl get service web`

```
kubernetes@kubernetes-VirtualBox:~$ kubectl get service web
NAME   TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
web    NodePort   10.104.117.81   <none>        8080:30676/TCP   37s
```

6. `minikube service web --url`

```
kubernetes@kubernetes-VirtualBox:~$ minikube service web --url
http://172.17.0.2:30676
```

Create ingress resource

1. Create file:

service/networking/example-ingress.yaml

```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
```

2. `kubectl apply -f` https://k8s.io/examples/service/networking/example-ingress.yaml

```
kubernetes@kubernetes-VirtualBox:~/service/networking$ kubectl apply -f example-
ingress.yaml
ingress.networking.k8s.io/example-ingress created
```

3. `kubectl get ingress`

```
kubernetes@kubernetes-VirtualBox:~/service/networking$ kubectl get ingress
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.2
2+; use networking.k8s.io/v1 Ingress
NAME              CLASS    HOSTS              ADDRESS      PORTS   AGE
example-ingress   <none>   hello-world.info   172.17.0.2   80      92s
```

4. Then I had to add this line on the bottom of /etc/hosts file
   172.17.0.2 hello-world.info
5. `curl hello-world.info`

```
kubernetes@kubernetes-VirtualBox:/etc$ curl hello-world.info
Hello, world!
Version: 1.0.0
Hostname: web-79d88c97d6-52nk9
```

Create second deployment

1. `kubectl create deployment web2 --image=gcr.io/google-samples/hello-app:2.0`

```
kubernetes@kubernetes-VirtualBox:/etc$ kubectl create deployment web2 --image=gc
r.io/google-samples/hello-app:2.0
deployment.apps/web2 created
```

2. `kubectl expose deployment web2 --port=8080 --type=NodePort`

```
kubernetes@kubernetes-VirtualBox:/etc$ kubectl expose deployment web2 --port=808
0 --type=NodePort
service/web2 exposed
```

3. Update paths in example-ingress.yaml

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

4. `kubectl apply -f example-ingress.yaml`

```
kubernetes@kubernetes-VirtualBox:~/service/networking$ kubectl apply -f example-
ingress.yaml
ingress.networking.k8s.io/example-ingress configured
```

Test deployments

1. `curl hello-world.info`

```
kubernetes@kubernetes-VirtualBox:~/service/networking$ curl hello-world.info
Hello, world!
Version: 1.0.0
Hostname: web-79d88c97d6-52nk9
```

2. `curl hello-world.info/v2`

```
kubernetes@kubernetes-VirtualBox:~/service/networking$ curl hello-world.info/v2
Hello, world!
Version: 2.0.0
Hostname: web2-5d47994f45-nxkrs
```